

58/pob

SPECIFICATION

TAG PRIVACY PROTECTION METHOD, TAG DEVICE,
BACKEND APPARATUS, UPDATER, UPDATE SOLICITOR,
5 PROGRAMS THEREFOR AND RECORD MEDIUM
CARRYING SUCH PROGRAMS IN STORAGE

TECHNICAL FIELD

[0001] The present invention relates to a tag technology incorporating
10 information security technology, in particular, to a method of protecting tag
privacy against acquisition of a user privacy information from information
delivered from a tag device, a tag device, a backend apparatus, an updater, an
update solicitor, programs therefor and a record medium carrying such
programs in storage.

15 BACKGROUND ART

[0002] Recently an automatic tag identification system such as RFID (Radio
Frequency Identification) is increasingly introduced. The system comprises an
information record medium of a small size referred to as "tag device", a
reading machine referred to as "reader" and a database server referred to as
20 "backend apparatus", and is utilized in controlling the distribution of articles.
A summary of this technology will be given below.

[Processing by tag device]

In a basic automatic tag identification system, each tag device has a
tag ID information which is inherent thereto (for example, a tag ID as
25 prescribed by Auto-ID center of MIT comprises a manufacturer code, a goods
code indicating the variety of goods and a serial number indicating the
number of a particular one of goods) in storage therein. The tag device is

applied to articles or the like, and the tag ID information which is inherent to each tag device is transmitted by radio communication to a reader installed in a store or the like.

[0003] [Processing by reader]

5 A reader reads tag ID information from a tag device through radio communication, and sends the tag ID information to a backend apparatus to solicit an acquisition of products distribution information.

[Processing by backend apparatus]

10 A backend apparatus controls a database for ID's from each tag device and for products distribution information or the like. And the backend apparatus retrieves products distribution information or the like in the database using tag ID information transmitted from the reader as a key and transmits a result of retrieval to the reader.

[0004] [Issues in basic automatic tag identification system]

15 However, in the basic automatic tag identification system, anyone who is in possession of a reader can read tag ID information, and accordingly, there has been a risk that information of articles under control may leak through eavesdropped tag ID information.

20 As regards this, non-patent literature 2 discloses a method in which a tag device delivers a hash value to a reader.

According to this method, the tag device initially transmits a hash value $H(id \parallel r)$ for a bit combination of ID information id and a random number r to the reader, which sends them to the backend apparatus. The backend apparatus forms a bit combination of the received random number r and each id' stored in the database, and determines its hash value $H(id' \parallel r)$. Then it verifies whether or not the determined hash value $H(id' \parallel r)$ matches with the received hash value $H(id \parallel r)$, and transmits products distribution

25

information or the like which corresponds to the matched id' to the reader. In this manner, a leakage of the tag ID information to a third party can be prevented. It is to be noted that $H(*)$ means a processing which applies a hash function H to $*$.

- 5 [0005] In a method disclosed in Patent Applications No. 2003-111342 and No. 2003-113798 which are not yet made open, a privileged ID which makes tag ID information confidential is employed to prevent a leakage of tag ID information to a third party. Specifically, in these techniques, a privileged ID is stored in a tag device, and a client apparatus which has read the privileged
- 10 ID solicits a security server apparatus on a network to decrypt the privileged ID. In response to the solicitation, the security server apparatus responds with a plain text tag ID information which is a decrypted result for the privileged ID after it has confirmed that the client is a regular client apparatus. In this manner, a leakage of tag ID information to a third party can be prevented.
- 15 non-patent literature 1: EPC global, Inc., "EPC global", [online], [retrieved September 9, 2004], internet <<http://www.epcglobalinc.org/>> .
- non-patent literature 2: Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, Daniel W. Engels, Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems, First International Conference on Security
- 20 in Pervasive Computing.

DISCLOSURE OF THE INVENTION

ISSUES TO BE SOLVED BY THE INVENTION

- [0006] However, with a conventional method, it is possible that a distribution process may be traced utilizing information which is delivered
- 25 from a tag device.

Specifically, with the method disclosed in non-patent literature 2, for example, a hash value $H(id \parallel r)$ which is transmitted from a tag device to a

reader is simply a random number to a third party who does not know id. A random number r is generated each time a communication occurs between the tag device and the reader, and accordingly, the hash value $H(id | r)$ changes from communication to communication. Accordingly, an attacker normally cannot gain a knowledge of an association between the hash value $H(id | r)$ which is eavesdropped from the tag device and a hash value $H(id | r_i)$ in a history of past communications. However, if an attacker could acquire ID information id as by tampering with the tag device, he could then calculate a hash value $H(id | r_i)$ from the random number r_i in the history of past communications (if he knows a hash function H). By verifying whether or not the calculated value coincides with the hash value (corresponding to the random number r_i) in the history of past communications, the attacker can know whether or not the history of the communications is one which corresponds to the acquired ID, and thus can trace a distribution process of the tag device by collecting the history of communications corresponding to this ID.

[0007] Also in the method disclosed in Patent Application No.2003-111342 or the like, for example, because the radio tag device always returns a same privileged ID, the attacker can trace the distribution process of the tag device by tracing the privileged ID if he cannot decrypt ID in plain text.

The present invention has been made in view of such aspect, and has for its object the provision of a technology which is capable of preventing a tracing of the distribution process of tag device by a third party.

MEANS TO SOLVE ISSUES

[0008] To overcome above issues, a tag device is arranged to have privileged information for tag ID information in storage, and it is updated by overwriting at a given opportunity. This makes it difficult for an attacker to

realize an association between information delivered in the past from a tag device and an updated privileged information, leading to a difficulty to trace the distribution process of the tag device.

By way of example, according to a first invention, a confidential value which corresponds to respective tag ID information is stored in a confidential value memory of respective tag devices. In response to an access from a reader, an output section of the tag device delivers a tag output information which corresponds to the confidential value in the confidential value memory. The tag device includes a first calculator, which reads out at least part of elements of the confidential value in the confidential value memory, and which applies a first function, the inverse image of which is difficult to obtain, and the confidential value in the confidential value memory is updated by overwriting with a result of such calculation. Since the confidential value in the confidential value memory is updated by overwriting, if an attacker acquires a confidential value stored in the confidential value memory as by tampering, the updated confidential value does not correspond to information which is transmitted from the tag device before the update. Since the update is achieved by applying the first function F1, the inverse image of which is difficult to obtain, it is difficult to obtain the confidential value before the update from a confidential value which prevails at a certain point in time. Accordingly, the attacker cannot know a correspondence between the tag device and the history of communications,

[0009] According to a second invention, for example, in an updater which is provided externally of a tag device, privileged ID information stored in the tag device is updated at a given opportunity into a new privileged ID information, an association of which with the original privileged ID information is difficult to follow. When the privileged ID information is

updated in this manner, an attacker cannot know a correspondence between a privileged ID information which is delivered from the tag device to backend apparatus before the update and new privileged ID information after the update. Accordingly, the attacker cannot know the correspondence between
5 the tag device and the history of communications.

EFFECTS OF THE INVENTION

[0010] As mentioned above, according to the present invention, a third party cannot know the correspondence between a tag device and a history of communications, and accordingly, a tracing of a distribution process of the
10 tag device by a third party can be prevented.

BREIF DESCRIPTION OF THE DRAWINGS

[0011] [Fig. 1] A is a block diagram illustrating an entire automatic tag identification system according to a first mode of carrying out the invention; B and C are block diagrams illustrating schematic arrangements of a tag
15 device and a backend apparatus, respectively;

[Fig. 2] is an illustration of an overall arrangement of an automatic tag identification system of an embodiment 1;

[Fig. 3] is a flow chart for describing processing in the embodiment
1;

20 [Fig. 4] is an illustration of an overall arrangement of an automatic tag identification system according to an embodiment 2;

[Fig. 5] is an illustration of an overall arrangement of an automatic tag identification system according to an embodiment 3;

[Fig. 6] is a flow chart for describing processing by a backend
25 apparatus of the embodiment 3;

[Fig. 7] is an illustration of an overall arrangement of an automatic tag identification system according to an embodiment 4;

[Fig. 8] is a flow chart for describing processing by the backend apparatus of the embodiment 4;

[Fig. 9] is an illustration of an overall arrangement of an automatic tag identification system of an embodiment 5;

5 [Fig. 10] A is a flow chart for describing processing by a tag device of the embodiment 5, and B is a flow chart for describing processing by a backend apparatus of the present embodiment;

[Fig. 11] is an illustration of an overall arrangement of an automatic tag identification system of an embodiment 6;

10 [Fig. 12] is a flow chart for describing processing in the embodiment 6;

[Fig. 13] is an illustration of an overall arrangement of an automatic tag identification system of an embodiment 7;

15 [Fig. 14] is a flow chart for describing processing in the embodiment 7;

[Fig. 15] is an illustration of an overall arrangement of an automatic tag identification system of an embodiment 8;

20 [Fig. 16] A is an illustration of data which is stored in the confidential value memory of a tag device, and B is an illustration of data stored in a database memory of a backend apparatus;

[Fig. 17] is a flow chart for describing processing in the embodiment 8;

[Fig. 18] is a flow chart for describing processing in the embodiment 8;

25 [Fig. 19] is an illustration of an overall arrangement of an automatic tag identification system of an embodiment 9;

[Fig. 20] A is an illustration of data stored in the confidential value

memory of a tag device, and B is an illustration of data stored in a database memory of a backend apparatus;

[Fig. 21] is an illustration of an overall arrangement of an automatic tag identification system of an embodiment 10;

5 [Fig. 22] is a flow chart for describing processing by a tag device of the embodiment 10;

[Fig. 23] is a flow chart for describing processing by a backend apparatus of the embodiment 10;

10 [Fig. 24] is an illustration of an overall arrangement of an automatic tag identification system of an embodiment 11;

[Fig. 25] is a flow chart for describing processing by a tag device of the embodiment 11;

[Fig. 26] is a flow chart for describing part of processing by a backend apparatus of the embodiment 11;

15 [Fig. 27] is a flow chart for describing a processing by a tag device of an embodiment 12;

[Fig. 28] is a block diagram showing a schematic arrangement of a second mode for carrying out the invention;

20 [Fig. 29] is a conceptual view illustrating an overall arrangement of an updater system of an embodiment 14;

[Fig. 30] is a block diagram showing a functional arrangement of an updater system of the embodiment 14;

[Fig. 31] is a flow chart for describing a processing procedure of the embodiment 14;

25 [Fig. 32] is a block diagram showing a functional arrangement of an updater system of an embodiment 15;

[Fig. 33] is a flow chart for describing a processing procedure of the

embodiment 15;

[Fig. 34] is a block diagram showing a functional arrangement of an updater system of an embodiment 16;

[Fig. 35] is a flow chart for describing a processing procedure in the
5 embodiment 16;

[Fig. 36] is a block diagram illustrating a functional arrangement of an updater system of an embodiment 17;

[Fig. 37] is a flow chart for describing a processing procedure of the embodiment 17;

10 [Fig. 38] is a conceptual view showing an overall arrangement of an updater system of an embodiment 18;

[Fig. 39] is a block diagram showing a functional arrangement of an updater system of the embodiment 18;

[Fig. 40] is a flow chart for describing a processing procedure of the
15 embodiment 18;

[Fig. 41] is a block diagram showing a functional arrangement of an updater system of an embodiment 19;

[Fig. 42] is a flow chart for describing a processing procedure of the embodiment 19;

20 [Fig. 43] is a block diagram showing a functional arrangement of an updater system of an embodiment 20;

[Fig. 44] is a flow chart for describing a processing procedure of the embodiment 20;

[Fig. 45] is a block diagram showing a functional arrangement of an
25 updater system of an embodiment 21;

[Fig. 46] is a block diagram showing a functional arrangement of an updater system of an embodiment 22;

[Fig. 47] is a conceptual view illustrating an overall arrangement of an updater system of an embodiment 23;

[Fig. 48] is an illustration of a functional arrangement of the updater system of the embodiment 23;

5 [Fig. 49] is a flow chart for describing a processing procedure of the embodiment 23;

[Fig. 50] is a flow chart for describing a processing procedure of the embodiment 23;

10 [Fig. 51] is an illustration of a functional arrangement of a security server apparatus of an embodiment 24;

[Fig. 52] is an illustration of a format used in the embodiment 24;

[Fig. 53] is a flow chart for describing a processing procedure of the security server apparatus of the embodiment 24;

15 [Fig. 54] is an illustration of a functional arrangement of an updater system of an embodiment 25;

[Fig. 55] is an illustration of a functional arrangement of an updater system of the embodiment 25;

[Fig. 56] is a flow chart for describing a processing procedure of the embodiment 25;

20 [Fig. 57] is a flow chart for describing a processing procedure of the embodiment 25; and

[Fig. 58] is an illustration of a functional arrangement of a tag device in an embodiment 26.

Description of characters

25 [0012] 1 automatic tag identification system
10 tag device
11 confidential value memory

	12	first calculator
	13	second calculator
	14	output section
	20	reader
5	30	backend apparatus
	31	database memory
	32	input section
	33	calculator
	34	comparator
10	35	read-out section
	40	network
	1500	updater system
	1510	tag device
	1511	confidential value memory
15	1512	read/write section
	1513	output section
	1514	input section
	1560	security server apparatus
	1561	input section
20	1562	updater
	1563	output section

BEST MODES FOR CARRYING OUT THE INVENTION

[0013] Several modes for carrying out the present invention will be described below with reference to the drawings.

25 [First mode]

<arrangement>

Fig. 1A is a block diagram illustrating an entire automatic tag

identification system 1 according to a first mode. B and C are block diagrams illustrating schematic arrangements of a tag device 10 and a backend apparatus 30, respectively.

As illustrated in Fig. 1A, an automatic tag identification system 1 of the present mode comprises a tag device 10, a reader 20, and a backend apparatus 30 which is connected to the reader 20 through a network 40.

[0014] As illustrated in Fig. 1B, the tag device 10 of the present mode comprises a confidential value memory 11 in which a confidential value corresponding to each tag ID information is stored, a first calculator 12 for applying a first function F1, the inverse image of which is difficult to obtain, a second calculator 13 for applying a second function F2 which disturbs a relationship between elements of a domain of definition and its mapping and an output section 14 for delivering tag output information which corresponds to a confidential value in the confidential value memory 11 to the backend apparatus 30.

As illustrated in Fig. 1C, the backend apparatus 30 of the present mode comprises a database memory 31 containing respective tag ID information and corresponding confidential values in a manner relating to each other, an input section 32 for accepting an input of tag output information, a calculator 33 for applying the first function F1 and the second function F2, a comparator 34 for comparing a result of calculation in the calculator 33 against the tag output information, and a read-out section 35 for extracting information from the database memory 31.

[0015] <processing by the tag device 10>

When the tag device 10 receives a read-out demand from the reader 20, the second calculator 13 of the tag device 10 initially reads out a confidential value from of the confidential value memory 11, and generates

the tag output information which is obtained by applying the second function F2 thereto. This tag output information is delivered to the output section 14 where it is delivered (by either radio or wire communication) to the backend apparatus 30. Subsequently, the first calculator 12 reads out at least part of the elements of the confidential value from the confidential value memory 11, applies the first function F1 thereto, and updates the confidential value in the confidential value memory 11 by overwriting with a result of such calculation. While the confidential value in the confidential value memory 11 is updated by overwriting after the tag output information has been generated, an arrangement may be such that the tag output information is generated after the confidential value in the confidential value memory 11 is updated by the overwriting.

[0016] <processing by the reader 20>

The reader 20 accepts an input of the tag output information which is delivered from the tag device 10 to the backend apparatus 30, and transmits it to the backend apparatus 30 through the network 40.

<processing by the backend apparatus 30>

The input section 32 of the backend apparatus 30 accepts an input of the tag output information which is transmitted from the reader 20. This triggers the calculator 33 to apply the first function F1 used in the tag device 10 some number of times to elements which represent at least part of the confidential value in the database memory 31 and also to apply the second function F2 used in the tag device 10. Result of calculation in the calculator 33 are successively compared against the tag output information in the comparator 34, and when a matching therebetween is found, the read-out section 35 extracts the tag ID information which is related to the confidential value which corresponds to the matched result of calculation from the

database memory 31.

[0017] [Embodiment 1]

Fig. 2 illustrates an overall arrangement of an automatic tag identification system 100 in an embodiment 1 according to the first mode, and
5 Fig. 3 is a flow chart for describing processing in the embodiment 1.

Referring to these Figures, the functional arrangement and a method of processing in the embodiment 1 will be described below.

<arrangement>

As illustrated in Fig. 2, the automatic tag identification system 100 of
10 the embodiment 1 comprises a tag device 110, a reader 120 and a backend apparatus 130 which is connected to the reader 120 through a network 140 so as to be capable of communication therewith. While in Fig. 2, only one tag device 110 is shown for purpose for simplifying the description, it should be noted that more tag devices exist in actuality. In addition, while one reader
15 120 and one backend device 130 are shown in Fig. 2, more readers 120 and backend apparatus 130 may be used to construct the present system.

[0018] <tag device>

The tag 110 device in this example comprises a confidential value memory 111, a hash calculator 112 (equivalent to “second calculator”), a hash
20 calculator 113 (equivalent to “first calculator”), an interface 114 (equivalent to “output section”), and a controller 115 including a memory 115a.

It is to be noted that the confidential value memory 111 and the memory 115a are memories capable of read/write operation such as EEPROM (Electrically Erasable and Programmable Read Only Memory), FeRAM
25 (Ferroelectric Random Access Memory), a flash memory, NV (Nonvolatile) RAM or the like, for example.

[0019] The hash calculator 112 and the hash calculator 113 are integrated

circuits constructed to apply one way functions or hash functions $G, H: \{0, 1\}^* \rightarrow \{0, 1\}^L$ to input values, for example, and to deliver results obtained. It is to be noted that $\{0, 1\}^*$ represents a set of all binary series and $\{0, 1\}^L$ represents a set of binary series having an L-bit length. Such hash functions G, H can be illustrated by SHA-1, MD5 or the like. It is to be noted that the hash function H is equivalent to “a first function $F1$, the inverse image of which is difficult to obtain”, and the hash function G is equivalent to “a second function $F2$ which disturbs a relationship between elements of a domain of definition and its mapping”. It is also to be noted that the controller 115 is an integrated circuit constructed so as to control processing of the entire tag device 110.

[0020] The interface 114 is a hardware which delivers data to the reader 120 by radio or wire communication, for example. Specifically, the interface 114 comprises an encoder/decoder circuit which performs an encoding/decoding using NRZ code, Manchester encoding, Miller code, single polarity RZ encoding or the like, a modulation/demodulation circuit which performs a modulation/demodulation by using the ASK (Amplitude Shift Keying), PSK (Phase Shift Keying), FSK (Frequency Shift Keying) or the like, and an antenna such as a dipole antenna, a microstrip antenna, a loop antenna or a cored coil to perform a transmission and reception of a signal using frequency in a low frequency band or ISM band (Industry Science Medical band). It is to be noted that the communication system utilizes the electromagnetic induction system or radio wave system.

[0021] The hash calculator 112 and the hash calculator 113 are electrically connected to the confidential value memory 111, and the hash calculator 112 is electrically connected to the interface 114 (equivalent to “output section”). While omitted from illustration in this Figure, the controller 115 is electrically

connected to various portions of the tag device 110.

<reader>

The reader 120 in this example comprises a products distribution information memory 121, an interface 122, a communication section 123, a
5 memory 124a, and a controller 124.

[0022] The products distribution information memory 121 comprises a magnetic recorder such as a hard disc unit, flexible disc, or the like, an optical disc unit such as DVD-RAM (Random Access Memory), CD-R (Recordable) /RW (ReWritable) or the like, a magneto-optical recorder such as MO
10 (Magneto-Optical disc), a semiconductor memory such as EEP-ROM (Electrically Erasable and Programmable-Read Only Memory), a flash memory or the like, for example. The interface 122 is a hardware which is similar to the example of the interface 114, for example. The communication section 123 comprises a LAN card, modem, a terminal adapter or the like, for
15 example, and the controller 124 comprises CPU (Central Processing Unit) of CISC (Complex Instruction Set Computer) type, RICS (Reduced Instruction Set Computer) type or the like and including the memory 124a.

[0023] The interface 122 and the products distribution information memory 121 are electrically connected to the communication section 123, and while
20 omitted from illustration in this Figure, the controller 124 is electrically connected to various portions of the reader 120.

<backend apparatus>

The backend apparatus 130 in this example comprises a database memory 131, a communication section 132 (equivalent to “input section”), a
25 hash calculator 133 (equivalent to “third calculator”), a comparator 134, a read/write section 135 (equivalent to “read-out section”), a memory 136a and a controller 136. Specifically, the backend apparatus 130 is constructed by the

execution of a given program by a known computer of Neumann type including CPU, RAM, ROM (Read Only Memory), an external memory such as a magnetic recorder, an optical disc unit or the like, a LAN card, a modem, a terminal adapter or the like, which are connected together by buses. The

- 5 CPU reads out a program stored in the RAM and executes a processing operation in accordance therewith to implement processing functions which are indicated below.

[0024] <pre-processing>

- A given program is installed into the backend apparatus 130 so that
10 the hash calculator 133 of the backend apparatus 130 can use the same hash functions G and H as contained in the tag device 110.

- A confidential value $s_{k,1}$ (equivalent to “first confidential value”) which corresponds to each tag ID information id_k ($k \in \{1, \dots, m\}$ where k corresponds to each tag device and m a total number of tag devices) is stored,
15 one each, in the confidential value memory 111 of each tag device 110. The confidential value $s_{k,1}$ is a pseudo-random number $s_{k,1} \in \{0, 1\}^L$ which is generated by a random number generator (not shown) which is provided externally of the tag device 110, for example, in accordance with pseudo-random number generating algorithm which is based on a computer
20 theory which uses a one-way hash function such as SHA-1 or the like. It is assumed that the random numbers $s_{k,1}$ which are stored in different tag devices are mutually non-coincident. A confidential value $s_{n,1}$ (equivalent to “second confidential value”, $n \in \{1, \dots, m\}$ and n corresponds to k) corresponding to each tag device n , tag ID information id_n and data such as
25 products distribution information $data_n$ are stored in the database memory 131 of the backend apparatus 130 in a manner relating to each other.

[0025] <processing by tag device>

Processings which occur when the tag device 110 is read by the reader 120 at an i -th run (where i is a natural number) will be described below. It is to be noted that the processing by the tag device 110 takes place under the control of the controller 115, and data which is necessary for the control is sequentially read and written from or into the memory 115a.

Initially, in the hash calculator 112, a confidential value $s_{k,i}$ (equivalent to “first confidential value”) is read from the confidential value memory 111 (step S1), and its hash value or tag output information $G(s_{k,i})$ is generated (step S2). This tag output information $G(s_{k,i})$ is sent to the interface 114 where it is transmitted to the reader 120 by either radio or wire communication (step S3).

[0026] Next, in the hash calculator 113, a hash value $s_{k,i+1}=H(s_{k,i})$ of the confidential value $s_{k,i}$ which is read from the confidential value memory 111 is calculated (step S4), and the hash value $s_{k,i+1}$ is saved in the confidential value memory 111 by overwriting as a new confidential value $s_{k,i+1}$ (equivalent to “new first confidential value”). (The confidential value $s_{k,i}$ in the confidential value memory 111 is erased, and instead the confidential value $s_{k,i+1}$ is stored: step S5.) It is to be understood that $H(*)$ implies a processing operation which applies the hash function H to $*$.

<processing by the reader>

The processing by the reader 120 takes place under the control of the controller 124, and data which is necessary for the control is sequentially read from or written into the memory 124a.

[0027] Initially, the interface 122 of the reader 120 receives tag output information $G(s_{k,i})$ transmitted from the tag device 110 (step S6), and sends it to the communication section 123. The communication section 123 extracts products distribution information pd (such as a store code where the reader

120 is installed, for example) from the products distribution information memory 121 (step S7), and transmits the products distribution information pd together with the tag output information $G(s_{k,i})$ to the backend apparatus 130 through the network 140 (step S8).

5 <processing by the backend apparatus>

Processing by the backend apparatus 130 takes place under the control of the controller 136, and data which is necessary for the control is sequentially read and written from and into the memory 136a.

[0028] Initially, the communication section 132 of the backend apparatus
10 130 receives the products distribution information pd and tag output information $G(s_{k,i})$ transmitted by the reader 120 (accepts inputs: step S9). The received products distribution information pd and tag output information $G(s_{k,i})$ are stored in the memory 136a. Next, the controller 136 enters 1 for n, which is then stored in the memory 136a (step S10). The controller 136 then
15 causes the hash calculator 133 to extract a confidential value $s_{n,1}$ from the database memory 131 while referring to n value in the memory 136a (step S11). The controller 136 then enters 0 for j, which is then stored in the memory 136a (step S12). The controller 136 then refers to j value in the memory 136a and causes the hash calculator 133 to calculate a hash value
20 $G(H^j(s_{n,1}))$ (equivalent to “result of calculation in the third calculator”) (step S13). It is to be noted that $H^j(s_{n,1})$ implies applying the hash function H to the confidential value $s_{n,1}$ j times. $H^0(s_{n,1})$ implies $s_{n,1}$.

[0029] Subsequently, in the comparator 134, the hash value $G(H^j(s_{n,1}))$ is acquired from the hash calculator 133 and the tag output information $G(s_{k,i})$ is
25 acquired from the memory 136a, and the comparator 134 compare them against each other (step S14).

When these values do not match (step S15), the controller 136 enters

$j+1$ for j in the memory 136a (step S16), and determines whether or not j has exceeded a given maximum value j_{\max} (step S17). When j is equal to or less than the maximum value j_{\max} , the controller 136 causes the processings which start with the step S13 to be executed again, and when j exceeds the
5 maximum value j_{\max} , it determines whether or not n in the memory 136a is equal to m (step S18). If $n=m$ does not apply, the controller 136 causes $n+1$ to be stored for n in the memory 136a (step S19), and causes the processings which start with step S11 to be executed again, and terminate the processings if $n=m$. It is to be noted that such processing is equivalent to executing the
10 processings in the hash calculator 133 and the comparator 134 again by changing the value of at least one of n and j under the control of the controller 136 when the tag output information $G(s_{k,i})$ - and the hash value $G(H^j(s_{n,i}))$ do not match.

[0030] On the other hand, in the event the tag output information $G(s_{k,i})$
15 and the hash value $G(H^j(s_{n,i}))$ match (step S15), the controller 136 sends the confidential value $s_{n,i}$ corresponding to the matched hash value $G(H^j(s_{n,i}))$ to the read/write section 135, which then extracts tag ID information id_n and data such as products distribution information $data_n$ which are related to the confidential value $s_{n,i}$ corresponding to the matched hash value $G(H^j(s_{n,i}))$
20 from the database memory 131 and sends them to the communication section 132 (step S20). The read/write section 135 receives the products distribution information pd from the memory 136a and writes it into the database memory 131 in a manner relating to the confidential value $s_{n,i}$ (step S20).

[0031] The tag ID information id_n and the data $data_n$ sent to the
25 communication section 132 are transmitted to the reader 120 through the network 140 (step S21), and are received by the communication section 123 of the reader 120 to be delivered (step S22).

<features of embodiment 1>

[impossibility of tracing]

In the embodiment 1 of the present mode, the hash value $G(s_{k,i})$ is used in the communication as a tag output information. On account of the incapability of recognizing a hash value, this hash value $G(s_{k,i})$ appears to be a random number simply for an attacker who does not know a confidential value. Accordingly, this attacker cannot know whether or not $G(s_{k,i})$ and $G(s_{k,i+1})$ are delivered from the same tag device 110, and therefore, cannot trace the distribution process of the tag device 110.

10 [0032] [forward security]

In the embodiment 1 of the present mode, the confidential value in the confidential value memory 111 which is used in the communication is arranged to be updated in accordance with a hash function H . If the tag device 110 is subjected to a tampering to leak the confidential value $s_{k,i}$, the attacker cannot determine the past confidential value $s_{k,i-\Delta i}$ from the confidential value $s_{k,i}$ because of the one-way nature of the hash function. Accordingly, if the confidential value $s_{k,i}$ leaks, the attacker cannot find a correspondence between the acquired confidential value $s_{k,i}$ and the history of communications, and hence cannot trace the tag device 110.

20 [0033] [traceability]

On the other hand, on account of the difficulty of collisions between the hash functions G and H (the property that hash values for different values hardly assume a same value), the backend apparatus 130 which knows the confidential value $s_{n,i}$ can trace the distribution process of the tag device 110.

25 [efficiency]

Because only the calculation of hash functions constructs communication data, the scale of a circuit which is incorporated in the tag

device 110 is small as compared with a conventional method of generating random numbers, and thus lends itself to an application for which a low price is demanded.

[0034] The hash value $H^j(s_{n, i})$ which is calculated at step S13 in the backend apparatus 130 may be recorded in the memory 136 to be utilized at a step S13 of the next loop. Specifically, using the recorded $H^j(s_{n, i})$, a hash value $H^{j+1}(s_{n, i})$ may be determined by $H(H^j(s_{n, i}))$, and this value may be stored in the memory 136a. In this instance, the number of times the hash calculation is made in the hash calculator 133 can be reduced, allowing the calculation efficiency of the backend apparatus 130 to be improved.

[0035] [Embodiment 2]

An embodiment 2 is a modification of the embodiment 1, and differs from the embodiment 1 only in respect of the fact that the tag device additionally carries tag ID information id_k (equivalent to “first proper value w_k ”) to update the confidential value $s_{k, i}$ according to $s_{k, i+1}=H(s_{k, i} \parallel id_k)$. In the description to follow, only distinctions over the embodiment 1 will be described.

Fig. 4 illustrates an overall arrangement of an automatic tag identification system 200 according to the embodiment 2. It is to be noted that in this Figure, parts which are common with the embodiment 1 are designated by common characters as used in the embodiment 1. Referring to this Figure, the functional arrangement and a processing method of the embodiment 2 will be described below.

[0036] <pre-processing>

A distinction over the embodiment 1 lies in the fact that tag ID information id_k and a corresponding confidential value $s_{k, i}$ are stored in a confidential value memory 211 of a tag device 210. A backend apparatus 130

includes a database memory 131 in which a confidential value $s_{n, 1}$, tag ID information id_n and data such as products distribution information $data_n$ which correspond to each tag device n are stored in a manner relating them to each other, and this tag ID information id_n is equivalent to “second proper value w_n ”.

< processing by tag device >

A difference with respect to the embodiment 1 lies only in the processing at step S4. Specifically, in place of the processing at the step S4 in the embodiment 1, a hash calculator 213 (equivalent to “first calculator”) extracts a confidential value $s_{k, i}$ and tag ID information id_k from a confidential value memory 211 to calculate $s_{k, i+1} = H(s_{k, i} | id_k)$. It is to be noted that $\alpha | \beta$ implies a bit combination of α and β . This result of calculation is overwritten into the confidential value memory 211 as a confidential value $s_{k, i+1}$.

[0037] <processing by reader >

This remains to be the same as in the embodiment 1.

<processing by backend apparatus >

A difference with respect to the embodiment 1 lies only in the processings at steps S11, S13, and S14. Specifically, in the embodiment 2, in place of step S11, a hash calculator 233 (equivalent to “third calculator”) of a backend apparatus 230 extracts a confidential value $s_{n, 1}$ and corresponding tag ID information id_n from the database memory 131.

[0038] Then, in the similar manner as in the embodiment 1, the controller 136 enters 0 for j , and stores it in the memory 136a (step S12). Subsequently, in place of step S13, the hash calculator 233 calculates a hash value $G(I^j(n))$. Here, a definition is made that $I^j(n) = s_{n, 1}$ ($j=0$), $I^j(n) = H(I^{j-1}(n) | id_n)$ ($j \geq 1$). Thus the hash calculator 233 determines $I^j(n)$ recurrently from the confidential

value $s_{n, i}$ and the corresponding tag ID information id_n , and calculates its hash value $G(I^j(n))$. This recurrent calculation is implemented by temporarily storing each $I^j(n)$ ($j' \in \{1, \dots, j-1\}$) appearing in the calculation process in the memory 136a to be used in the calculation of next $I^{j+1}(n)$. $I^j(n)$ which is
5 obtained when calculating the hash value $G(I^j(n))$ may be saved in the memory 136a at least until the next hash value $G(I^{j+1}(n))$ is calculated. In this manner, $I^j(n)$ which is once obtained can be utilized in the calculation of $I^{j+1}(n)=H(I^j(n) \mid id_n)$ which is used in determining the next hash value $G(I^{j+1}(n))$, allowing an improved efficiency of calculation.

10 [0039] Subsequently, in place of step S14, a comparator 134 acquires the hash value $G(I^j(n))$ from the hash calculator 233 and tag output information $G(s_{k, i})$ from the memory 136a and compare them against each other. Subsequently, the processings at step S15 and subsequent steps are executed in the similar manner as in the embodiment 1.

15 As described above, in the embodiment 2, the confidential value $s_{k, i}$ in the confidential value memory 211 of the tag device 210 is updated by a calculation $s_{k, i+1}=H(s_{k, i} \mid id_k)$. In this manner, a situation that updated contents of confidential values which correspond to different tag ID information id_k may become semi-permanently coincident can be prevented. Specifically,
20 when the same hash function is applied to different confidential values or the like, it is possible that result of these calculations may become coincident at a certain point in time (collision). However, even in such instance, tag ID information id_k which corresponds to each confidential value $s_{k, i}$ is different, and hence a next confidential value which is calculated according to $s_{k, i+1}=H(s_{k, i} \mid id_k)$ cannot be the same. This represents an effect that cannot be
25 obtain when a confidential value is updated according to $s_{k, i+1}=H(s_{k, i})$.

[0040] While tag ID information id_k and id_n have been used as the first

proper value w_k and the second proper value w_n in the embodiment 2, other information which corresponds to each tag ID information may be used as a proper value.

[Embodiment 3]

5 This represents a modification of the embodiment 1, and the difference with respect to the embodiment 1 exists only in recording a calculated value $G(H^j(s_{n,1}))$ ($j=0, \dots, j_{\max}$) which is previously calculated in the backend apparatus. Only a distinction over the embodiment 1 will be described below.

10 [0041] Fig. 5 is an illustration of an overall arrangement of an automatic tag identification system 300 according to an embodiment 3. In this Figure, parts which are common to the embodiment 1 are designated by common characters as used in the embodiment 1. Fig. 6 is a flow chart for describing processings by a backend apparatus 330 in the embodiment 3. A functional
15 arrangement and the processing method of the embodiment 3 will be described below with reference to these Figures.

<pre- processing>

20 Storing a result of calculation $G(H^j(s_{n,1}))$ ($j=0, \dots, j_{\max}$) which is previously calculated by the hash calculator 133 in a database memory 331 of the backend apparatus 330 in a manner relating it to the confidential value $s_{n,1}$ represents a sole distinction with the respect to the first mode.

[0042] <processing by tag device/processing by reader>

 These remain to be similar to the embodiment 1.

<processing by backend apparatus>

25 Initially, the backend apparatus 330 receives the products distribution information pd and tag output information $G(s_{k,i})$ transmitted from the reader 120 by means of a communication section 132 (step S31). The received

products distribution information pd and tag output information $G(s_{k,i})$ are stored in a memory 136a. A controller 136 then enters 1 for n , and stores it in the memory 136a (step S32). The controller 136 then enters 0 for j , and stores it in the memory 136a (step S33). The controller 136 extracts a result of calculation $G(H^j(s_{n,1}))$ stored in a database memory 331 while referring to values of n and j in the memory 136a (step S34).

[0043] Then, a comparator 134 compares this result of calculation $G(H^j(s_{n,1}))$ against the tag output information $G(s_{k,i})$ which is extracted from the memory 136a (step S35).

In the event these values do not match (step S36), the controller 136 enters $j+1$ for j in the memory 136a (step S37) and determines whether or not j has exceeded the given maximum value j_{\max} (step S38). If j is less than the maximum value j_{\max} , the controller 136 causes processings at step S34 and subsequent steps to be re-executed, and when j exceeds the maximum value j_{\max} , it determines whether or not n in the memory 136a is equal to m (step S39). If $n=m$ does not apply, the controller 136 stores $n \leftarrow n+1$ (making $n+1$ to be a new n) in the memory 136a (step S40), causes processings at step 33 and subsequent steps to be re-executed, and terminates the processing operations for $n=m$. This operation is equivalent to re-executing the processings in the hash calculator 133 and the comparator 134 by changing the value of at least one of n and j under the control of the controller 136 when the tag output information $G(s_{k,i})$ - and the hash value $G(H^j(s_{n,1}))$ do not match.

[0044] On the other hand, when the tag output information $G(s_{k,i})$ and the hash value $G(H^j(s_{n,1}))$ match (step S36), the controller sends the confidential value $s_{n,i}$ corresponding to the matched result of calculation $G(H^j(s_{n,1}))$ to the read/write section 135, which extracts the tag ID information id_n and data $data_n$ such as the products distribution information or the like which are

related to the confidential value $s_{n,1}$ which corresponds to the matched hash value $G(H^j(s_{n,1}))$ from the database memory 331 and sends them to the communication section 132 (step S40). The read/write section 135 receives the products distribution information pd from the memory 136a, and writes
5 this products distribution information pd into the database memory 131 by relating it to the confidential value $s_{n,1}$ (step S40). The tag ID information id_n and the data $data_n$ which are sent to the communication section 132 are transmitted to the reader 120 through the network 140 (step S41).

[0045] As described above, in the embodiment 3, an arrangement is made to
10 store the result of calculation $G(H^j(s_{n,1}))$ which is previously calculated in the database memory 331. Consequently, the amount of a processing in the backend apparatus 330 can be reduced as compared with an arrangement in which $G(H^j(s_{n,1}))$ is calculated for each comparing processing.

[Embodiment 4]

15 An embodiment 4 is a modification of the embodiment 1, and differs from the embodiment 1 only in an arrangement that information specifying a number of times a confidential value is updated is transmitted from a tag device, and the number of times the confidential value is updated is used in a backend apparatus for purpose of processing. Only a distinction over the
20 embodiment 1 will be described below.

[0046] Fig. 7 is an illustration of an overall arrangement of an automatic tag identification system 400 of the an embodiment 4. In this Figure, parts common to the embodiment 1 are designated by common characters as used in the embodiment 1. Fig. 8 is a flow chart for describing processing by a
25 backend apparatus 430 of the embodiment 4. A functional arrangement and a processing method of the embodiment 4 will be described below with reference to these Figures.

<arrangement of tag device>

A distinction over the embodiment 1 resides in a provision of a counter 416 in a tag device 410 which counts a number of times rn a confidential value is updated.

5 [0047] <processing by tag device>

A distinction over the embodiment 1 resides in only an arrangement which stores a number of times rn a confidential value $s_{k,i}$ is updated as counted by a counter 416 in a confidential value memory 411 of a tag device 410 in addition to the confidential value $s_{k,i}$ and an arrangement by which
10 information which specifies the number of times rn the update is performed is transmitted to the reader 120 through a hash calculator 112 and an interface 114 (equivalent to “output section”).

<processing by reader>

A distinction over the embodiment 1 resides only in an arrangement
15 in which an interface 122 also receives information specifying a number of times rn the update is performed and a communication section 123 transmits information specifying the number of times rn the update is performed to the backend apparatus 430 through the network 140.

[0048] <processing by backend apparatus>

20 Initially, the communication section 132 of the backend apparatus 330 receives information specifying rn , products distribution information pd and tag output information $G(s_{k,i})$ which are transmitted from the reader 120 (step S50). Information specifying rn , the products distribution information pd and the tag output information $G(s_{k,i})$ which have been received are stored in
25 the memory 136a. Then the controller 136 enters 1 for n , and stores it in the memory 136a (step S51). The controller 136 then causes a hash calculator 433 to extract a confidential value $s_{n,i}$ from the database memory 131 while

referring to values of n and j in the memory 136a (step S52), and causes the hash function H to be applied thereto m times and also causes the hash function G to be applied subsequently, thus allowing a hash value $G(H^j(s_{n,1}))$ ($j=mn$) to be calculated (step S53).

- 5 [0049] Then the comparator 134 acquires the hash value $G(H^j(s_{n,1}))$ from the hash calculator 133 and the tag output information $G(s_{k,i})$ from the memory 136a and compare them against each other (step S54).

In the event these values do not match (step S55), the controller 136 determines whether or not n in the memory 136a is equal to m (step D56). If
10 $n=m$ does not apply, the controller 136 stores $n \leftarrow n+1$ (making $n+1$ to be a new n) in the memory 136a (step S57), causes processings at step S52 and subsequent steps to be re-executed and terminates the processing if $n=m$. It is to be noted that this processing is equivalent to re-executing the processings in the hash calculator 433 and the comparator 134 by changing the value of n
15 when the hash value $G(H^j(s_{n,1}))$ and the tag output information $G(s_{k,i})$ do not match.

[0050] On the other hand, if the tag output information $G(s_{k,i})$ and the hash value $G(s_{k,i})$ match (step S55), the controller sends the confidential value $s_{n,1}$ corresponding to the matched hash value $G(s_{k,i})$ to the read/write section 135,
20 which then extracts tag ID information id_n and data $data_n$ such as products distribution information which are related to the confidential value $s_{n,1}$ which corresponds to the matched hash value $G(H^j(s_{n,1}))$ from the database memory 131 and sends them to the communication section (step S58). The read/write section 135 receives products distribution information pd from the
25 memory 136a, and then writes this products distribution information pd into the database memory 131 in a manner relating it to the confidential value $s_{n,1}$ (step S59). Tag ID information id_n and data $data_n$ which are sent to the

communication section 132 are transmitted to the reader 120 through the network 140 (step S59).

[0051] As described above, in the embodiment 4, an arrangement is made so that the tag device 410 transmits rn and the backend apparatus 430 uses this rn to calculate the hash value $G(H^m(s_{n,1}))$ for purpose of a comparing processing. In this manner, a comparing processing by the backend apparatus 430 takes place only once for each $s_{n,1}$, allowing the amount of the processing required to be reduced.

[Embodiment 5]

10 An embodiment 5 is a modification of the embodiment 1, and differs from the embodiment 1 only in respect of performing an updating/comparison of the confidential value using a secret key encrypted function in place of a hash function. Only a distinction over the embodiment 1 will be described below.

15 [0052] Fig. 9 is an illustration of an overall arrangement of an automatic tag identification system 500 of the embodiment 5. It is to be noted that in this Figure, parts which are common to the embodiment 1 are designated by common characters as used in the embodiment 1. Fig. 10A is a flow chart for describing processing by a tag device 510 in the embodiment 5, and Fig. 10B
20 is a flow chart for describing processing by a backend apparatus 530 in the embodiment 5. A functional arrangement and a processing method of the present embodiment will be described below with reference to these Figures.

<pre-processing>

In the embodiment 5, a tag device 510 is provided with a key
25 memory 515, and a backend apparatus 530 is provided with a key memory 536, each storing common keys KG and KH , respectively. In the tag device 510, encrypted function calculators 512 and 513 are substituted for the hash

calculators 112 and 113 of the first embodiment, and in the backend apparatus 530, an encrypted function calculator 533 is substituted for the hash calculator 133. The encrypted function calculators 512, 513, and 533 are constructed to enable a calculation with a common key encryption function E such as AES, Camellia or the like in place of the hash function. In the embodiment 5, the common key encryption function E which uses the common KH is equivalent to “a first function F1, the inverse image of which is difficult to obtain”, and the common key encryption function E which uses the common keys KG is equivalent to “a second function F2 which disturbs a relationship between elements of a domain of definition and its mapping”. Thus, the first function F1 and the second function F2 in this example represent the same common key encryption function, to which different common keys are applied.

[0053] What is mentioned above represents a distinction over the embodiment 1.

<processing by tag device>

Initially, an encrypted function calculator 512 (equivalent to “second calculator”) extracts a confidential value $s_{k,i}$ from a confidential value memory 111 (step S61), extracts a common key KG from a key memory 515, and applies a common key encryption function E to the secret key $s_{k,i}$ with the common key ($E_{KG}(s_{k,i})$: step S62). An encrypted text $E_{KG}(s_{k,i})$ which is calculated is transmitted as the tag output information $E_{KG}(s_{k,i})$ from the interface 114 to the reader 120 through radio or wire communication (step S63).

[0054] Then, in an encrypted function calculator 513 (equivalent to “first calculator”), the common key KH is extracted from the key memory 515, the confidential value $s_{k,i}$ is extracted from the confidential value memory 111, and the common key encryption function E is applied to the confidential value

$s_{k,i}$ with the common key KH (step 64), and a result of this calculation is saved by overwriting as a new confidential value $s_{i+1}=E_{KH}(s_{k,i})$ in the confidential value memory 111 (step 65).

<processing by reader>

5 This remains to be similar as in the embodiment 1,

[0055] <processing by backend apparatus>

Initially, the backend apparatus 530 receives products distribution information pd and tag output information $E_{KG}(s_{k,i})$ transmitted by the reader 120 by the communication section 132 (step S70). Received products

10 distribution information pd and tag output information $E_{KG}(s_{k,i})$ are stored in the memory 136a. Then, the controller 136 enters 1 for n, and stores it in the memory 136a (step S71). The controller 136 then causes the encrypted function calculator 533 (equivalent to “third calculator”) to extract the confidential value $s_{n,1}$ from the database 131 while referring to the value n in

15 the memory 136a (step S72). The controller 136 then enters 0 for j, and stores it in the memory 136a (step S73). The controller 136 causes the encrypted function calculator 533 to calculate an encrypted text $E_{KG}(E_{KH}^j(s_{n,1}))$ (equivalent to “result of a calculation in the third calculator”) while referring to the value of j in the memory 136a (step S74). It should be noted that $E_{KH}^j(s_{n,1})$

20 ($s_{n,1}$) implies applying a common key encryption function E to the confidential value $s_{n,1}$ j times using the common key KH. Then the comparator 134 acquires the encrypted text $E_{KG}(E_{KH}^j(s_{n,1}))$ from the hash calculator 133 and acquires tag output information $E_{KG}(s_{k,i})$ from the memory 136a, and compare them against each other (step S75).

25 [0056] In the event these values do not match (step S76), the controller 136 enters j+1 for j in the memory 136a (step S77), and determines whether not j has exceeded a given maximum value j_{max} (step S78). If it is found that j is

equal to or less than the maximum value j_{\max} , the controller 136 causes processings at step S74 and subsequent steps to be re-executed and if j has exceeded the maximum value j_{\max} , determines whether or not n in the memory 136a is equal to m (step S79). If $n=m$ does not apply, the controller 136 saves
5 $n \leftarrow n+1$ (making $n+1$ to be a new n) in the memory 136a (step S80), causes the processings at step 72 and subsequent steps to be re-executed and terminates the processing operation if $n=m$. It is to be noted that this processing is equivalent to re-executing the processings in the encrypted function calculator 533 and the comparator 134 by changing the value of at least one of n and j under the control of the controller 136 when tag output
10 information $E_{KG}(s_{k,i})$ and the encrypted text $E_{KG}(E_{KH}^j(s_{n,1}))$ do not match. [0057] On the other hand, if tag output information $E_{KG}(s_{k,i})$ and the encrypted text $E_{KG}(E_{KH}^j(s_{n,1}))$ match (step S76), the controller 136 sends the confidential value $s_{n,1}$ which corresponds to the matched encrypted text E_{KG}
15 $(E_{KH}^j(s_{n,1}))$ to the read/write section 135, which then extracts the tag ID information id_n and data $data_n$ such as products distribution information which are related to the confidential value $s_{n,1}$ which corresponds to the matched encrypted text $E_{KG}(E_{KH}^j(s_{n,1}))$ from the database memory 131, and sends them to the communication section 132 (step S81). The read/write section 135
20 receives products distribution information pd from the memory 136a, and writes the products distribution information pd into the database memory 131 by relating it with the confidential value $s_{n,1}$ (step S81). Tag ID information id_n and data $data_n$ sent to the communication section 132 are transmitted to the reader 120 through the network 140 (step S82).
25 [0058] It is to be noted that the encrypted text $E_{KH}^j(s_{n,1})$ which is calculated at step S74 in the backend apparatus 530 may be recorded in the memory 136a to be utilized at the step S74 of the next loop. Specifically, using $E_{KH}^j(s_{n,1})$

s_i) which is recorded, an encrypted text $E^{j+1}_{KH}(s_{n,i})$ may be determined according to $E_{KH}(E^j_{KH}(s_{n,i}))$, and this value may be stored in the memory 136a again. In this instance, a number of times an encrypted calculation is performed in the encrypted function calculator 533 can be reduced, improving the efficiency of the calculation by the backend apparatus 530. Alternatively, $E^j_{KH}(s_{n,i})$ ($j \in \{1, \dots, j_{\max}\}$) may be previously calculated in the backend apparatus 530 and stored in the memory 136a to be utilized at step S74. Again, the efficiency of calculation in the backend apparatus 530 can be improved.

[0059] While the embodiment 5 represents an example in which processings are performed by using the common key encryption function E which uses the common key KH as “the first function $F1$, the inverse image of which is difficult to obtain” and using the common key encryption function E which uses the common key KG as “the second function $F2$ which disturbs a relationship between elements of the domain of definition and its mapping”, processings may be performed by using a hash function for one of the first function $F1$ and the second function $F2$. Also in the embodiments 1 to 4 or embodiments 6 to 11 which will be described later, processings may be performed by using a common key encryption function E which uses a common key KH or KG for at least one of the first function $F1$ and the second function $F2$.

[0060] In this manner, in the embodiment 5, an arrangement is made to update a confidential value $s_{k,i}$ using a common key encryption function. As a consequence, if the confidential value $s_{k,i}$ leaks from the tag device 510, it is impossible for an attacker to trace a distribution process of the tag device 10 on the basis of the confidential value $s_{k,i}$ and a history of communications. Since there is no need to provide a random number generator circuit in the tag device 510, a cost required for the tag device 510 can be reduced. In addition,

if a common key encryption function which is lighter (requiring a lesser amount of calculations) than a hash function could be used, the amount of processings in the tag device 510 and backend apparatus 530 can be reduced.
[0061] [Embodiment 6]

5 An embodiment 6 is a modification of the embodiment 1 and differs from the embodiment 1 in that a hash value of a bit combination of a confidential value $s_{k,i}$ and a first proper value w_k which is inherent to each tag is used as the tag output information.

Fig. 11 is an illustration of an overall arrangement of an automatic
10 tag identification system 600 of the embodiment 6, and Fig. 12 is a flow chart for describing processing in the embodiment 6. It is to be noted that in Fig. 11, parts which are common to the embodiment 1 are designated by common characters as used in the embodiment 1. A functional arrangement and a processing method of the embodiment 6 will be described below with
15 reference to these Figures.

[0062] <pre-processing>

Differences over the embodiment 1 reside in an arrangement that a confidential value $s_{k,i}$ corresponding to each tag ID information id_k (equivalent to “first confidential value”) and a proper value w_k (equivalent to
20 “the first proper value”) are stored in a confidential value memory 611 of each tag device 610 and an arrangement in which each tag ID information id_n ($n \in \{1, \dots, m\}$) as well as a confidential value $s_{n,i}$ (equivalent to “second confidential value”), a proper value w_n (equivalent to “second proper value”) and data $data_n$ such as products distribution information which correspond
25 thereto are stored in a database memory 631 of a backend apparatus 630 in a manner relating them to each other. As a proper value, tag ID information may be utilized, for example.

[0063] <processing by tag device>

In the following, a processing which occurs when the tag device 610 is read by a reader 620 during an i -th run (i being a natural number) will be described.

5 Initially, a hash calculator 612 extracts a confidential value $s_{k,i}$ and a proper value w_k from the confidential value memory 611 (step S101), and calculates tag output information $G(s_{k,i} | w_k)$ by applying a hash function G to a bit combination of the confidential value $s_{k,i}$ and the proper value w_k (step S102). The interface 114 transmits this tag output information $G(s_{k,i} | w_k)$ to
10 the reader 120 by either radio or wire communication (step S103).

[0064] Then a hash calculator 113 calculates a hash value $H(s_{k,i})$ by applying the hash function H to confidential value $s_{k,i}$ which is extracted from the confidential value memory 611 (step S104), and overwrites the confidential value $s_{k,i}$ in the confidential value memory 611 by a new
15 confidential value $s_{k,i+1}$ which is the hash function $H(s_{k,i})$ (the confidential value $s_{k,i}$ in the confidential value memory 611 is erased and the confidential value $s_{k,i+1}$ is saved instead: step S105).

<processing by reader>

The reader 120 receives at its interface 122 tag output information
20 $G(s_{k,i} | w_k)$ which is transmitted from the tag device 610 (step S106) and sends it to the communication section 123. The communication section 123 extracts products distribution information pd from products distribution information memory 121 (step S107) and transmits the products distribution information pd and the hash value $G(s_{k,i} | w_k)$ to the backend apparatus 630
25 through the network 140 (step S108).

[0065] <processing by backend apparatus>

The backend apparatus 630 receives the products distribution

information pd and the tag output information $G(s_{k,i} | w_k)$ which are transmitted from the reader 120 at its communication section 132 (or accepts inputs: step S109). The received products distribution information and tag output information $G(s_{k,i} | w_k)$ are stored in a memory 136a.

5 Then the controller 136 enters 0 for parameters j and n , and store them in the memory 136a (step S10). The controller 136 then refers to j and n in the memory 136 and causes a hash calculator 633 (equivalent to “third calculator”) to calculate a hash value $G(H^j(s_{n,1}) | w_n)$ using a set of second confidential value $s_{n,i}$ and second proper value w_n which are extracted from
10 the database memory 631 (step S111). It is to be noted this $H^j(s_{n,1})$ may be calculated beforehand and stored in the database memory 631. In this instance, a calculation load of the backend apparatus 630 can be alleviated.
[0066] Then, a comparator 134 acquires a hash value $G(H^j(s_{n,1}) | w_n)$ from the hash calculator 633 and acquires tag output information $G(s_{k,i} | w_k)$ from
15 the memory 136a and compare them against each other (step S112).

 In the event these values do not match (step S113), the controller 136 enters $j+1$ for j in the memory 136a (step S114), and determines whether or not j has exceeded a given maximum value j_{max} (step S115). If j is equal to or less than the maximum value j_{max} , it returns to the processing at step S111, but
20 if j exceeds the maximum value j_{max} , the controller 136 enters $n+1$ for n and 0 for j in the memory 136a (step S116), and determines whether n has exceeded a given maximum value n_{max} (step S117). If n is equal to or less than the maximum n_{max} , it returns to the processing at step S111, but if n exceeds the maximum value n_{max} , an error termination results (step S118).

25 [0067] On the other hand, if a determination at step S113 reveals that the tag output information $G(s_{k,i} | w_k)$ and the hash value $G(H^j(s_{n,1}) | w_n)$ match, the controller 136 applies this value of n to the read/write section 135, which uses

this n to extract id_n and $data_n$ which are related to the confidential value $s_{n, i}$ and the proper value w_n which correspond to the matched hash value $G(H^j(s_{n, i}) | w_n)$ from the database memory 631, and send these to the communication section 132. The read/write section 135 also receives the products distribution information pd from the memory 136a, and writes this products distribution information pd into the database memory 631 in a manner relating it to the confidential value $s_{n, i}$ and the proper value w_n which correspond to the matched hash value $G(H^j(s_{n, i}) | w_n)$ (step S119).

[0068] id_n and $data_n$ which are sent to the commutation section 132 are transmitted to the reader 120 through the network 140, and are received by the communication section 123 of the reader 120 to be delivered (step S121).

<features of embodiment 6>

In the embodiment 6, tag output information $G(s_{k, i} | w_k)$ delivered from each tag device 610 represents a hash value of a bit combination of the confidential value $s_{k, i}$ and the proper value w_k which is inherent to each tag device 610. The confidential value $s_{k, i}$ of each tag device is successively updated by the hash value $H(s_{k, i})$. If tag output information $G(s_{k, i} | w_k)$ becomes identical between tag devices (occurrence of a collision), because the proper value w_k differs between tag devices, this collision can be eliminated with a high probability due to the difficulty of a collision occurring between hash functions if the confidential value $s_{k, i}$ of each tag device is updated. In this manner, a collision between tag output information $G(s_{k, i} | w_k)$ between tag devices 610 can be prevented from occurring in a continued manner, thus preventing a failure of the backend apparatus 630 to identify tag ID information uniquely from the tag output information $G(s_{k, i} | w_k)$.

[0069] [Embodiment 7]

An embodiment 7 is a modification of the embodiment 6, and differs

from the embodiment 6 in that tag devices shares a confidential value in common. A distinction over the embodiment 1 and embodiment 6 will be principally described below.

Fig. 13 is an illustration of an overall arrangement of an automatic tag identification system 700 of the embodiment 7. It is to be noted that in this Figure, parts which are common to the embodiment 1 are designated by common characters as used in the embodiment 1. Fig. 14 is a flow chart for describing processing in the embodiment 7. A functional arrangement and a processing method of the embodiment 7 will be described below with reference to these Figures.

[0070] <pre-processing>

For each ID (id_k ($k=1, \dots, m$)) which corresponds to each tag device 710, a single random number $s_1 \in \{0, 1\}^t$ is generated, and is stored as a confidential value s_1 (which is an initial value of s_i and is equivalent to “first confidential value”) in the confidential value memory 711 of each tag device 710. For each tag ID information (id_k ($k=1, \dots, m$)) which corresponds to each tag device 710, a proper value w_k which is inherent to each is generated, and is stored in the confidential value memory 711 of the respective tag device 710.

[0071] The confidential value s_1 which is the same as the confidential value s_1 stored in each tag device 710 is stored in a database memory 731 of a backend apparatus 730 as “second confidential value”. Each proper value w_n is also stored the database memory 731 in a manner relating it to tag ID information id_n and $data_n$ such as products distribution information or the like of the corresponding tag device 710.

In addition, a hash value $s_{j+2} = H^{j+1}(s_1)$ ($j=0, \dots, j_{\max}$) of the confidential value s_1 which is common to each tag device 710 is calculated by

a hash calculator 736 of the backend apparatus 730. Each calculated hash value s_{j+2} is stored in the database memory 731.

[0072] <processing by tag device>

5 In the following, a processing which takes place during an i -th run when the tag device is read by a reader 720 will be described.

Initially, a hash calculator 712 extracts a confidential value s_i and a proper value w_k from the confidential value memory 711 (step S131), and calculates a tag output information $G(s_i | w_k)$ which is a hash value of a bit combination of the confidential value s_i and the proper value w_k (step S132).

10 The interface 114 transmits this tag output information $G(s_i | w_k)$ to the reader 120 (step S133).

[0073] A hash calculator 113 then calculates a hash value $H(s_i)$ of the confidential value s_i which is extracted from the confidential value memory 711 (step S134) and the confidential value s_i in the confidential value memory 15 711 is overwritten by the hash value $H(s_i)$ as a new confidential value s_{i+1} (step S135).

<processing by reader>

This remains to be similar to the embodiment 1 (steps S136 ~ S138).

<processing by backend apparatus>

20 The backend apparatus 730 receives the products distribution information pd and tag output information $G(s_i | w_k)$ transmitted from the reader 120 at its communication section 132 (step S139). The received products distribution information pd and tag output information $G(s_i | w_k)$ are stored in the memory 136a.

25 [0074] Then the controller 136 enters 0 for parameters j and n and store them in the memory 136a (step S140).

In a hash calculator 733 (equivalent to “third calculator”), a hash

value $G(s_{j+1} | w_n)$ is calculated using the proper value w_n and the confidential value s_1 which are extracted from the database memory 731 or the hash value s_{j+2} (which is calculated beforehand by the hash calculator 736) (step S141).

Then, a comparator 134 acquires the hash value $G(s_{j+1} | w_n)$

5 (equivalent to “result of calculation in the third calculator”) from the hash calculator 733 and acquires the tag output information $G(s_i | w_k)$ from the memory 136a, and compare them against each other (step S142).

[0075] In the event these values do not match (step S143), the controller 136 enters $j+1$ for j in the memory 136a (step S144), and determines whether
10 or not j has exceeded a given maximum value j_{\max} (step S145). If j is equal to or less than the maximum value j_{\max} , it returns to the processing at step S141, but if j has exceeded the maximum value j_{\max} , the controller substitutes $n+1$ for n and 0 for j in the memory 136a (step 146) and determines whether or not n has exceeded a given maximum value n_{\max} (step S147). If n is equal to or
15 less than the maximum value n_{\max} , it returns to the processing at step S141, but if n has exceeded the maximum value n_{\max} , an error termination results (step S148).

[0076] If the determination at step S143 reveals that the tag output information $G(s_i | w_k)$ and the hash value $G(s_{j+1} | w_n)$ match, the read/write
20 section 135 extracts id_n and $data_n$ which are related to the proper value w_n which corresponds to the matched hash value $G(s_{j+1} | w_n)$ from a database memory 731 and sends them to the communication section 132 under the control of the controller 136. The read/write section 135 also receives products distribution information pd from the communication section 132,
25 and writes the products distribution information pd into the database memory 731 in a manner relating it to the proper value w_n which corresponds to the matching hash value $G(s_{j+1} | w_n)$ (step S149).

[0077] id_n and $data_n$ which are sent to the communication section 132 are transmitted to the 120 through the network 140 (step S150), and are received by the communication section 123 of the reader 120 to be delivered (step S151).

5 <features of embodiment 7>

In the embodiment 7, the confidential value s_1 which is common to each tag device 710 is used. Accordingly, the confidential value s_{j+1} which is used in the processing at step S141 by the backend apparatus 730 can be used in common to each tag ID information id_n , whereby the amount of
10 calculations in the backend apparatus 730 can be drastically reduced, permitting an efficient retrieval.

[0078] Specifically, denoting the number of the tag devices 710 by m and the number of hashing operations in the backend apparatus 730 (the number of times the confidential value is updated for the tag device 710) by j , the
15 embodiment 1 required a number of hash operations which is equal to $2mj$. By contrast, in the embodiment 7, the number of hash operations can be suppressed to $mj+j$.

In addition, the tag device 710 delivers the number of times rn the confidential value s_1 is updated together with the tag output information $G(s_i | w_k)$, and if the number of the times it is updated rn is fed to the backend
20 apparatus 730 (see the embodiment 4), the number of hash operations in the backend apparatus 730 can be reduced down to $m+j$.

[0079] [Embodiment 8]

An embodiment 8 is a modification of the embodiment 1, and differs
25 from the embodiment 1 in that a combination of a plurality of elements is allotted as a value which is inherent to each tag device. In this manner, part of elements which are allotted to each tag device can be shared by a plurality of

tag devices, with consequence that a total amount of calculations which are required to recognize a tag device can be reduced.

Fig. 15 is an illustration of an overall arrangement of an automatic tag identification system 800 of the embodiment 8. In this Figure, parts which are common to the embodiment 1 are designated by like numerals as used in the embodiment 1. Fig. 16 A shows an example of data which are stored in a confidential value memory 811 of a tag device 810, and Fig, 16 B shows examples of data which are stored in a database memory 831 of a backend apparatus 830. In addition Fig. 17 and Fig. 18 are flow charts for describing processings in the embodiment 8.

[0080] A functional arrangement and a processing method of the embodiment 8 will be described below with reference to these Figures. It is to be noted that what is common to the embodiment 1 will be omitted from description.

<pre-processing>

By way of example, using a random number generator (not shown) or the like, a set of initial values of elements which are allotted to respective tag devices

$(b_{1,1,0}, \dots, b_{1,j,0}, \dots, b_{1,\rho,0}) \cdots (b_{u,1,0}, \dots, b_{u,j,0}, \dots, b_{u,\rho,0}) \cdots (b_{d,1,0}, \dots, b_{d,j,0}, \dots, b_{d,\rho,0})$

are generated. A set of elements within each “()” will be referred to as a sub-group $\alpha_u (u \in \{1, \dots, d\})$.

[0081] Here, j is a natural number ($j \in \{1, \dots, \rho\}$) which satisfies $1 \leq j \leq \rho$, and u is a natural number ($u \in \{1, \dots, d\}$) which satisfies $1 \leq u \leq d$. In the embodiment 8, a combination of a plurality of elements defines one

confidential value, and $d (d \geq 2)$ represents a number of elements which constitute one confidential value. m is a number equal to or greater than a total number of tag devices 810 (a total number of required confidential

values) and which satisfies a requirement that $m=\rho^d$ is a natural number.

Combinations of elements thus generated are allotted to respective tag devices 810. Specifically, one element is selected from each of d kinds of sub-groups α_u which constitutes together the set of initial values of above

5 mentioned elements, and selected d combinations of initial elements $f_{u,0}$

$(f_{1,0}, \dots, f_{u,0}, \dots, f_{d,0})$ are allotted to respective tag devices 810

$(f_{1,0} \in \{b_{1,1,0}, \dots, b_{1,q,0}, \dots, b_{1,p,0}\}, \dots, f_{u,0} \in \{b_{u,1,0}, \dots, b_{u,q,0}, \dots, b_{u,p,0}\}, \dots, f_{d,0} \in \{b_{d,1,0}, \dots, b_{d,q,0}, \dots, b_{d,p,0}\})$. It is to be noted that this allotment

is made so that a same combination does not occur for different tag devices

10 810, and a total of m kinds (a total number of tag devices 810) of

combinations $(f_{1,0}, \dots, f_{u,0}, \dots, f_{d,0})$ are allotted. Alternatively, a plurality of

combinations of initial elements $f_{u,0}$ may be related to a single tag device 810,

and in this instance, a total of m kinds or more (a total number of tag devices 810 or more) combinations of $(f_{1,0}, \dots, f_{u,0}, \dots, f_{d,0})$ are allotted. At least part of

15 elements which constitute each $(f_{1,0}, \dots, f_{u,0}, \dots, f_{d,0})$ is shared by a plurality of tag devices 810.

[0082] All the combinations $(f_{1,0}, \dots, f_{u,0}, \dots, f_{d,0})$ which are generated

(combinations of $d(d \geq 2)$ initial elements $f_{u,0} (u \in \{1, \dots, d\})$ are related to tag ID information id_n of each allotted tag device 810 and data $data_n$

20 corresponding to each tag device 810, and are stored in a database memory

831 of the backend apparatus 830. It is to be noted that n assumes a value

which corresponds to each tag device, and corresponds to a suffix k of tag

output information $a_{k,i}$ (to be described later) which is delivered from each tag device. In other words, the number of combinations of d initial elements $f_{u,0}$

25 which are stored in the database memory 831 is equal to the total number of

tag devices 810. Where a plurality of combinations of initial elements $f_{u,0}$ are related to a single tag device 810, the number of combinations of d initial

elements $f_{u,0}$ which are stored in the database memory 831 will be equal to or greater than the total number of tag devices 810.

[0083] Combinations of generated initial elements $(f_{1,0}, \dots, f_{u,0}, \dots, f_{d,0})$

(equivalent to “combinations each comprising $d(d \geq 2)$ elements e_{u,v_u}

5 $(u \in \{1, \dots, d\})$ and corresponding to respective tag ID information id_k ”

where v_u represents an integer equal to or greater than 0 and indicating the

number of times the element e_{u,v_u} is updated and the suffix v_u of the element

e_{u,v_u} represents v_u) are stored in the confidential value memory 811 of

respective allotted tag devices 810. In the description to follow, a combination

10 of initial elements which is stored in the confidential value memory 811 of

each tag device 810 is indicated by $(e_{1,0}, \dots, e_{u,0}, \dots, e_{d,0})$.

[0084] In the example of Fig. 16, an allotment of initial elements for $d=2$,

$\rho=3$, $m=9$ is shown.

As shown in Fig. 16 B, for this example, combinations 831aa of

15 initial elements $((f_{1,0}, f_{2,0}) (f_{1,0} \in \{b_{1,1,0}, b_{1,2,0}, b_{1,3,0}\},$

$f_{2,0} \in \{b_{2,1,0}, b_{2,2,0}, b_{2,3,0}\})$, tag ID information 831ab ($id_n (n \in \{1, \dots, 9\})$)

and data 831ac ($data_n (n \in \{1, \dots, 9\})$) are stored in the database memory 831

of the backend apparatus 830 in a manner relating to each other.

[0085] As shown in Fig. 16 A, one set of combinations of initial elements

20 811a $((e_{1,0}, e_{2,0})=(b_{1,2,0}, b_{2,2,0}))$ which corresponds to the tag ID information id

is stored in the confidential value memory 811 of the tag device 810. It is to

be noted that part of the element e_{u,v_u} which is stored in the confidential value

memory 811 is also stored in the confidential value memory of another tag

device as a corresponding element in another tag device.

25 <processing by tag device>

A processing which takes place when the tag device 810 is read by the reader 20 during an i -th run (i is a natural number) will be described

below.

[0086] Initially, in a hash calculator 812 (equivalent to “second calculator”),
d elements e_{u, v_u} are extracted from the confidential value memory 811 (step

5 (confidential value $s_{k, i}$) to calculate the tag output information $a_{k, i} = G(s_{k, i})$
(step S162) where k represents a value corresponding to each tag device and i
is a natural number indicating a number of times delivered from the output

section. It is to be noted that in the present embodiment, the confidential value
 $s_{k, i} = e_{l, v_l} \mid \cdots \mid e_{u, v_u} \mid \cdots \mid e_{d, v_d}$ and the tag output information

10 $a_{k, i} = G(e_{l, v_l} \mid \cdots \mid e_{u, v_u} \mid \cdots \mid e_{d, v_d})$ are used, but the sequence in which
bits of the respective elements e_{u, v_u} are disposed are not limited thereto.

[0087] The generated tag output information $a_{k, i}$ is sent to an interface 114,
which delivers the tag output information $a_{k, i}$ (step S163).

Subsequently, a hash calculator 813 (equivalent to “first calculator”)
15 extracts elements $e_{u', v_{u'}}$ ($u' \in \{1, \cdots, d\}$), which are at least part thereof, from
the confidential value memory 811, calculates a hash value $H(e_{u', v_{u'}})$ of the
extracted elements $e_{u', v_{u'}}$ (step S164) and saves by overwriting the hash value
 $H(e_{u', v_{u'}})$ as a new element $e_{u', v_{u'+1}}$ in the confidential value memory 811 (step
S165). It is to be understood that a method of selecting $u' \in \{1, \cdots, d\}$ may be

20 any desired one. By way of example, a method of selecting a different u' each
time the tag device 810 performs a communication, a method in which a
separate u' is selected at the time every element $e_{u', v_{u'}}$ has been updated for
one u' , a method of selecting two or more u' concurrently can be cited.

[0088] <processing by reader>

25 The reader 120 receives a tag output information $a_{k, i}$ transmitted
from the tag device 810 at its interface 122 (step S166), and sends it to the
communication section 123. The communication section 123 extracts

products distribution information pd from products distribution information memory 121 (step S167) and transmits the products distribution information pd and the tag output information $a_{k,i}$ to the backend apparatus 830 through the network 140 (step S168).

5 <processing by backend apparatus>

The tag output information $a_{k,i}$ and the products distribution information pd which are transmitted from the reader 120 are received by the communication section 132, and are stored in the memory 136a (step S169).

[0089] This triggers the controller 136 to substitute 1 for n and to store it in
10 the memory 136a (step S170), to select a combination of d w_u in the manner indicated below and to store the combination in the memory 136a (step S171).
 $(w_1, \dots, w_d) \in S_w = \{w_1, \dots, w_d \mid w_u \in [0, j_{\max}]\}$ (where $[\alpha, \beta]$ represents a set of integers equal to or greater than α and equal to or less than β .)

Then the controller 136 verifies, while referring to n and the
15 combination of d w_u in the memory 136a and also referring to a hash value memory 838, whether or not a hash value $H^{w_u}(f_{u,0})$ which is a result of applying w_u times the hash function H to d initial elements $f_{u,0}$ ($u \in \{1, \dots, d\}$) corresponding to the tag ID information id_n is stored (is already generated) in the hash value memory 838 (step S172). It is to be noted
20 that super-index wu in $H^{w_u}(f_{u,0})$ represents w_u .

[0090] In the event it is determined that there remain some of hash values $H^{w_u}(f_{u,0})$ corresponding to the tag ID information id_n which have not yet been calculated, a hash calculator 837 extracts initial elements $f_{u,0}$ corresponding to those of “the hash values $H^{w_u}(f_{u,0})$ corresponding to the tag ID information id_n
25 which have not yet been calculated” from the database memory 831, applies w_u times the hash function H to these initial elements $f_{u,0}$ to calculate the hash value $H^{w_u}(f_{u,0})$ (step S173). The calculated hash value $H^{w_u}(f_{u,0})$ is stored in

the hash value memory 838 (step S174), then returning to the processing at step S172.

[0091] On the other hand, if it is determined at step S172 that all of hash values $H^{wu}(f_{u,0})$ corresponding to the tag ID information id_n have been

5 generated, the controller 136 causes, while referring to n and combinations of d w_u in the memory 136a, a hash calculator 833 (equivalent to “third calculator”) to extract the hash values $H^{wu}(f_{u,0})$ which are obtained by applying w_u times the first function $F1$ to d initial elements $f_{u,0}$

($u \in \{1, \dots, d\}$) corresponding to the tag ID information id_n from the hash

10 value memory 838 (step S175) and to calculate a value c which is obtained by applying the hash value G to a bit combination value of these hash values

$H^{wu}(f_{u,0})$ (step 176). The calculated value c may be illustrated by

$c = G(H^{w1}(f_{1,0}) \mid \dots \mid H^{wu}(f_{u,0}) \mid \dots \mid H^{wd}(f_{d,0}))$, for example, but the bit disposition sequence of each hash value $H^{wu}(f_{u,0})$ is not limited thereto.

15 However, the sequence should correspond to the bit disposition sequence of each element e_{u,v_u} in the hash calculator 812 of the tag device 810.

[0092] Then, the comparator 134 reads the tag output information $a_{k,i}$ from the memory 136a, receives the calculated value c from the hash calculator 833 and compare them to determine if $c=a_{k,i}$ (step S177). In this example, the hash

20 value $c = G(H^{w1}(f_{1,0}) \mid \dots \mid H^{wu}(f_{u,0}) \mid \dots \mid H^{wd}(f_{d,0}))$ and the tag output information $a_{k,i}$ are compared against each other.

In the event it is determined that these do not match, the controller 136 determines whether or not all of d combination patterns

$(w_1, \dots, w_d) \in S_w$ have been selected while referring to the memory 136a

25 (step S178). If it is determined that there exists a combination pattern which has not yet been selected, the controller 136 selects a new combination

$(w_1, \dots, w_d) \in S_w$, stores it in the memory 136a (step S179) and causes the

processings at step S172 and subsequent steps to be executed for this new combination and n .

[0093] On the other hand, if it is determined at step S178 that all of combination patterns have been selected, the controller 136 determines whether or not $n=m$ while referring to n in the memory 136a (step S180). If it is determined that $n=m$ does not apply, the controller 136 updates n in the memory 136a by $n+1$ (step S181) and causes processings at step S172 and subsequent steps to be executed. On the other hand, if it is determined that $n=m$, an error termination of the processing results (step S182).

It is to be noted that the processings which take place at steps S172 ~ 181 are equivalent to re-executing the processings in the hash calculator 833 and the comparator 134 by changing the value of at least part of n and w_u under the control of the controller 136 when the tag output information $a_{k,i}$ and the calculated value c do not match.

[0094] On the other hand, if it is determined at step S177 that the hash value c and the tag output information $a_{k,i}$ match, the read/write section 135 selects tag ID information id_n which is related to the combination of a plurality of initial elements $f_{u,0}$ corresponding to the hash value c from the database memory 831 under the control of the controller 135, extracts the tag ID information id_n and its corresponding data $data_n$ and sends them to the communication section 132. The read/write section 135 receives the products distribution information pd from the memory 136a, and writes this products distribution information pd as data $data_n$ corresponding to the tag ID information id_n into the database memory 831 as an addition (step S183).

[0095] The tag ID information id_n and data $data_n$ which are sent to the communication section 132 are transmitted to the reader 120 through the network 140 (step S184), and are received by the communication section 123

of the reader 120 to be delivered (step S185).

<features of embodiment 8>

[efficiency]

To calculate the hash value c in the hash calculators 838 of the
5 backend apparatus 830, it is necessary to calculate a hash value $H^{wu}(f_{u,0})=f_{u,vu}$.
In the embodiment 8, each element $e_{u,vu}$ can be used in common by a plurality
of tag devices 810, and accordingly, if the hash value $H^{wu}(f_{u,0})=f_{u,vu}$ which is
calculated to calculate the hash value c corresponding to either one of the tag
devices 810 is stored in the hash value memory 838, this element $f_{u,vu}$ can
10 also be utilized in the calculation of hash values c corresponding to other tag
devices 810. In this manner, the number of tag devices 810 which can be
accommodated can be increased without increasing the number of hash values
 $H^{wu}(f_{u,0})$ which must be calculated. Specifically, initial elements which are
inherent to tag devices which are equal in number to ρ^d can be allotted by
15 using $d \cdot \rho$ elements.

[0096] Because communication data is constructed by the calculation of
hash functions alone, the scale of a circuit which is incorporated into the tag
device 810 is small in comparison to a conventional method which generates
random numbers, and this embodiment lends itself to an application which
20 demands a low cost.

[impossibility of tracing]

In the embodiment 8, tag output information $a_{k,i} = G(s_{k,i})$ is used in
the communication. Because of the incapability to identify a hash value, this
tag output information $a_{k,i} = G(s_{k,i})$ appears to be a mere random number to an
25 attacker who does not know a confidential value. Accordingly, the attacker
cannot know whether or not the tag output information $a_{k,i} = G(s_{k,i})$ and $a_{k,i+1} =$
 $G(s_{k,i+1})$ have been delivered from the same tag device 810, and therefore

cannot trace the distribution process of the tag device 810.

[0097] [forward security]

In the embodiment 8, an arrangement is used that the confidential value in the confidential value 811 which is used in the communication is updated by the hash function H. In addition, if the tag device 810 is tampered with to leak each element e_{u, v_u} , the attacker cannot determine a past element $e_{u, v_u - \Delta v_u}$ from the element e_{u, v_u} due to the one-way nature of the hash function. Accordingly, if each element e_{u, v_u} were leaked, the attacker cannot find a correspondence between each element e_{u, v_u} acquired and the history of communications, and hence cannot trace the tag device 810.

[0098] [traceability]

On the other hand, on account of the difficulty of a collision between the hash functions G and H (the property that hash values of different values can hardly assume a same value), the backend apparatus 830 which knows each element e_{u, v_u} is capable of tracing the distribution process of the tag device.

In the embodiment 8, the set of initial elements which are generated by the backend apparatus 830 is chosen to be as follows:

$(b_{1, 1, 0}, \dots, b_{1, j, 0}, \dots, b_{1, p, 0}) \cdots (b_{u, 1, 0}, \dots, b_{u, j, 0}, \dots, b_{u, p, 0}) \cdots (b_{d, 1, 0}, \dots, b_{d, j, 0}, \dots, b_{d, p, 0})$

Thus, p initial elements b are generated for each u ($u \in \{1, \dots, d\}$). However, the number of initial elements which are generated for each u ($u \in \{1, \dots, d\}$) may be different.

[0099] The hash value $H^{wu}(f_{u, 0})$ ($u \in \{1, \dots, d\}$) which is required in the processing at step S176 may be determined by the hash calculator 837 of the backend apparatus 830 at the pre-processing step, and stored in the hash value memory 838.

[Embodiment 9]

An embodiment 9 is a modification of the embodiment 8, and differs from the embodiment 8 in that a proper value which is inherent to each tag device is additionally stored in the confidential value memory of the tag device and the database memory of the backend apparatus, and a hash value
5 $a_{k,i} = G(s_{k,i})$ of a combination of bit trains including each element $e_{u,vu}$ and a proper value γ_k is used as tag output information. This allows a situation that a confidential value of a particular tag device be determined on the basis of elements $e_{u,vu}$ which are collected by tampering with other tag devices to trace the tag device to be prevented from occurring.

10 [0100] Only a distinction over the embodiment 8 will be described below while omitting a description for what are common with the embodiment 8.

Fig. 19 is an illustration of an overall arrangement of an automatic tag identification system 900 of the embodiment 9. Fig. 20 A illustrates examples of data which are stored in a confidential value memory 911 of a tag device 910, and Fig. 20 B shows examples of data which are stored in a
15 database memory 931 of a backend apparatus 930. It is to be noted that in Fig. 19, a functional arrangement which is common with the embodiment 1 is denoted by like characters as used in Fig. 2, and a functional arrangement which is common with the embodiment 8 is denoted by like characters as used in Fig. 15 without giving a description thereof. It is to be noted that
20 while Fig. 19 shows only one tag device 910, there are a plurality of tag device 910 in actuality.

[0101] A functional arrangement and a processing method of the embodiment 9 will be described below with reference to these Figures.

25 <pre-processing>

Differences over the embodiment 8 reside in that a proper value γ_k is additionally stored in the confidential value memory 911 of the tag device 910

and that a combination of d ($d \geq 2$) initial elements $f_{u,0}$ ($u \in \{1, \dots, d\}$), a proper value γ_k which is inherent to each tag device, and tag ID information id_n of each tag device (n assuming a value which corresponds to each tag device) are stored in the database memory 931 of the backend apparatus 930 in a manner relating these to each other. It is to be noted that proper values γ_k and γ_n are random values, for example.

[0102] In the example shown in Fig. 20, an allotment of combined proper values for $d=2$, $\rho=3$, $m=9$ is shown.

As shown in Fig. 20 B, in this example, a combination 931aa of initial elements $((f_{1,0}, f_{2,0}) (f_{1,0} \in \{b_{1,1,0}, b_{1,2,0}, b_{1,3,0}\}, f_{2,0} \in \{b_{2,1,0}, b_{2,2,0}, b_{2,3,0}\}))$, the tag ID information 931ab (id_n ($n \in \{1, \dots, 9\}$)), data 931ac ($data_n$ ($n \in \{1, \dots, 9\}$)), and a proper value 931ad which is inherent to each tag device (γ_k , $k \in \{1, \dots, 12\}$) are stored in the database memory 931 of the backend apparatus 930 in a manner relating these to each other. As shown in Fig. 20 A, a combination 911a of initial elements $((e_{1,0}, e_{2,0}) = (b_{1,2,0}, b_{2,2,0}))$ and a proper value 911b ($\gamma_k = \gamma_5$) are stored in the confidential value memory 911 of the tag device 910.

[0103] <processing by tag device>

A processing which occurs when the tag device 910 is read by the reader 120 during an i -th run (i being a natural number) will be described below.

Initially, a hash calculator 912 (equivalent to “second calculator”) extracts each element $e_{u,vu}$ and a proper value γ_k from the confidential value memory 911, and calculates the tag output information $a_{k,i} = G(s_{k,i})$ which is a hash value of a combination value (confidential value $s_{k,i}$) of bit trains including the extracted elements $e_{u,vu}$ and the proper value γ_k . In the embodiment 9, a confidential value and tag output information are chosen to

be $s_{k,i} = \gamma_k \mid e_{l,vl} \mid \cdots \mid e_{u,vu} \mid \cdots \mid e_{d,vd}$ and $a_{k,i} = G(\gamma_k \mid e_{l,vl} \mid \cdots \mid e_{u,vu} \mid \cdots \mid e_{d,vd})$.

[0104] Subsequently, the tag output information $a_{k,i}$ is delivered and the elements in the confidential value memory 911 are updated in the similar as in the embodiment 8.

<processing by reader>

This remains to be similar to the embodiment 8.

<processing by backend apparatus>

A distinction over the embodiment 8 resides in that in place of the processing at step S176 in the embodiment 8 (Fig. 18), a hash calculator 933 (equivalent to “third calculator”) reads a proper value γ_n from the database memory 931 and calculates a hash value c of a combination value of bit trains including the hash value $H^{wu}(f_{u,0})$ and the proper value γ_n . In this example, a hash value $c = G(\gamma_n \mid H^{w1}(f_{l,0}) \mid \cdots \mid H^{wu}(f_{u,0}) \mid \cdots \mid H^{wd}(f_{d,0}))$ is calculated. In other respects, the processing is similar to the embodiment 8.

[0105] <features of embodiment 9>

[impossibility of tracing]

In the embodiment 9, the tag output information $a_{k,i} = G(s_{k,i})$ which is a hash value of a combination of bit trains including elements $e_{u,vu}$ and the proper value γ_k is delivered from the tag device 910. The proper value γ_k is a value which is inherent to each tag device 910. Accordingly, if a certain tag device is tampered with, it is impossible to determine the past tag output information of a different tag device which shares the element $e_{u,vu}$ from the hash value of data which are stored in the first mentioned tag device.

Accordingly, an attacker cannot trace the different tag device.

[0106] [Embodiment 10]

An embodiment 10 is a modification of the embodiment 8, and

differs from the embodiment 8 in that t kinds ($t \geq 2$) of manifold values z are stored in a manifold value memory of a tag device, and a hash value $a_{k,i} = G(s_{k,i})$ of a bit combination value (confidential value $s_{k,i}$) of each element $e_{u,vu}$ extracted from a confidential value memory and either one of the manifold values z is used as tag output information, with the confidential value memory being updated once for communication which counts t times.

Only a distinction over the embodiment 8 will be described below while omitting a description for what is common with the embodiment 8.

[0107] Fig. 21 is an illustration of an overall arrangement of an automatic tag identification system 1000 of the embodiment 10. Fig. 22 is a flow chart for describing processing in a tag device 1010, and Fig. 23 is a flow chart for describing processing in a backend apparatus 1030. It is to be noted that in Fig. 21, functional arrangements which are common to the embodiment 1 and the embodiment 8 are designated by like characters as used in Figs. 2 and 15. It is also to be noted that while Fig. 21 shows only one tag device 1010, there exist a plurality of tag devices 1010 in actuality.

A functional arrangement and a processing method of the present embodiment will be described below with reference to these Figures.

[0108] <pro-processing>

Differences over the embodiment 8 reside in that t kinds ($t \geq 2$) of manifold values z are generated in a manifold value generator 1015 of the tag device 1010 and are stored in a manifold value memory 1016 (equivalent to “first manifold value memory”) and that t kinds ($t \geq 2$) of manifold values z which are shared by respective tag devices 1010 are stored in a database memory 1031 (equivalent to “second manifold value memory”) of the backend apparatus 1030.

A manifold value generator 1015 can be illustrated by a counter

which counts $z = 1 \cdots t$, a hash calculator which performs a calculation $z = H(\text{seed}, x)$, $x \in \{1, \cdots, t\}$, a hash calculator which performs a calculation $z = H^x(\text{seed})$, $x \in \{1, \cdots, t\}$ or the like. Here, seed represents an initial value.

In a description to follow, a manifold value z will be represented as $z = \pi(x)$

5 and count $x \in \{1, \cdots, t\}$. Preferably manifold values $z = \pi(x)$ which correspond to respective values of $x \in \{1, \cdots, t\}$ do not coincide.

[0109] It is to be noted that the generation and storage of the manifold value z need not be performed during the pre-processing, but may be performed when processing a communication with the tag device 1010 or when

10 processing a retrieval by the backend apparatus 1030.

<processing by tag device>

A processing which occurs when the tag device 1010 is read by the reader 120 during an i -th run (i being a natural number) will be describes below. It is to be noted that the initial value ($i=1$) of the count x is 1, and the
15 count x is saved in a memory 115a under the control of a controller 115.

[0110] Initially, a hash calculator 1012 (equivalent to “second calculator”) extracts each element e_{u, v_u} from a confidential value memory 1011 and extracts either one of manifold values z (which is $z = \pi(x)$ in this example) from a manifold value memory 1016 (step S191). The hash calculator 1012

20 calculates a hash function $a_{k, i} = G(s_{k, i})$ of a bit combination value (confidential value $s_{k, i}$) of each extracted element e_{u, v_u} and manifold value z as tag output information (step S192). In this example, a confidential value is chosen to be $s_{k, i} = e_{l, v_l} \mid \cdots \mid e_{u, v_u} \mid \cdots \mid e_{d, v_d} \mid z$, and tag output information is chosen to be $a_{k, i} = G(e_{l, v_l} \mid \cdots \mid e_{u, v_u} \mid \cdots \mid e_{d, v_d} \mid z)$. It is to be noted the bit

25 disposition sequence of each element e_{u, v_u} and the manifold value z and the number of manifold values z for bit combinations are not limited to this.

Since it is presumed that manifold values $z = \pi(x)$ corresponding to

$x \in \{1, \dots, t\}$ do not coincide, it follows that as long as elements in the confidential value 1011 are not updated, the manifold value z which is used by the hash calculator 1012 in generating tag output information $a_{k,i}$ varies from communication to communication.

- 5 [0111] The generated tag output information $a_{k,i}$ is sent to an interface 114, which delivers the tag output information $a_{k,i}$ (step S193).

Subsequently, a controller 115 performs an arithmetic operation $x \leftarrow x+1$ (count up) (step S194), and determines whether or not $x > t$ (step S195). If it is determined that $x > t$ does not apply, the processing in the tag device

- 10 1010 is terminated while maintaining the value x in the memory 115a.

On the other hand, if it is determined that $x > t$, the controller 115 changes the count x in the memory 115a to $x \leftarrow 1$ (step S196), and a hash calculator 1013 extracts at least part of elements $e_{u',vu'}$ ($u' \in \{1, \dots, d\}$) from the confidential value memory 1011 and calculates a hash value $H(e_{u',vu'})$ of
15 extracted element $e_{u',vu'}$ (step S197). The hash calculator 1013 overwrites this hash value $H(e_{u',vu'})$ into the confidential value memory 1011 as a new element $e_{u',vu'+1}$ (step S198). Any technique of selecting $u' \in \{1, \dots, d\}$ may be used.

[0112] <processing by reader>

- 20 This remains to be similar to the embodiment 8.

<processing by backend apparatus>

The tag output information $a_{k,i}$ and products distribution information pd which are transmitted from the reader 120 are received by a communication section 132 and stored in a memory 136a (step S201).

- 25 This triggers the controller 136 to substitute 1 for n to be stored in the memory 136a (step S202), to select a combination $d w_u$'s in the manner indicated below, and to store the combination in the memory 136a (step

S203).

[0113] $(w_1, \dots, w_d) \in S_w = \{w_1, \dots, w_d \mid w_u \in [0, j_{\max}]\}$

The controller 136 then refers to n and the combination of d w_u 's in the memory 136a and also refers to the hash value memory 838 to verify

5 whether or not hash values $H^{w_u}(f_{u,0})$ which are results of applying w_u times the hash function H to d initial elements $f_{u,0}$ ($u \in \{1, \dots, d\}$) corresponding to tag ID information id_n are stored in the hash value memory 838 (whether or not they have been generated) (step S204). It is to be noted that the super-index wu in $H^{wu}(f_{u,0})$ represents w_u .

10 [0114] If it is determined that there exist some of the hash values $H^{w_u}(f_{u,0})$ corresponding to the tag ID information id_n which have not yet been calculated, a hash calculator 837 extracts initial elements $f_{u,0}$ corresponding to “some of the hash values $H^{w_u}(f_{u,0})$ corresponding to tag ID information id_n which have not yet been calculated” from the database memory 1031, and
15 calculates the values $H^{w_u}(f_{u,0})$ by applying w_u times the hash function H to these initial elements $f_{u,0}$ (step S205). The calculated hash values $H^{w_u}(f_{u,0})$ are stored in the hash value memory 838 (step S206), then returning to the processing at the step S204.

[0115] On the other hand, if it is determined at the step S204 that all of hash
20 values $H^{w_u}(f_{u,0})$ corresponding the tag ID information id_n have been generated, the controller 136 causes, while referring to n and the combination of d w_u 's in the memory 136a, a hash calculator 1033 (equivalent to “third calculator”) to extract a hash value $H^{w_u}(f_{u,0})$ which is a result of applying w_u times the first function $F1$ to each of d initial elements $f_{u,0}$ ($u \in \{1, \dots, d\}$) corresponding to
25 the tag ID information id_n (step S207). The controller 136 sets up a count x' of 1 and stores it in the memory 136a (step S208) and extracts a manifold value $z=\pi(x')$ from the database memory 1031 to be fed to the hash calculator 1033.

The hash calculator 1033 then calculates a calculated value c by applying the hash value G to a bit combination value of the hash value $H^{wu}(f_{u,0})$ and the manifold value z (step S209). The calculated value c can be illustrated as $c = G(H^{wl}(f_{l,0}) \mid \cdots \mid H^{wu}(f_{u,0}) \mid \cdots \mid H^{wd}(f_{d,0}) \mid z)$, for example, but the bit disposition sequence of each hash values $H^{wu}(f_{u,0})$ and the manifold value z and the number of the manifold value z which are used in the bit combination are not limited to this. However, there is a requirement that the sequence thereof or the like should be related to the bit disposition sequence of elements in the hash calculator 1012 of the tag device 1010.

10 [0116] The comparator 134 then reads tag output information $a_{k,i}$ from the memory 136a, receives the calculated value c from the hash calculator 1033, and compare them to determine whether or not $c=a_{k,i}$ (step S210). In this example, the hash value $c = G(H^{wl}(f_{l,0}) \mid \cdots \mid H^{wu}(f_{u,0}) \mid \cdots \mid H^{wd}(f_{d,0}) \mid z)$ and the tag output information $a_{k,i}$ are compared against each other.

15 In the event it is determined that these do not match, the controller 136 determines whether or not x' in the memory 136a is t (step S211). If it is determined that $x'=t$ does not apply, the controller 136 updates x' in the memory 136a by $x'+1$ and then causes the processings at step S209 and subsequent steps to be executed (step S212). On the other hand, if it is
20 determined that $x'=t$, the controller 136 determines, by referring to the memory 136a, whether or not all of d combination patterns $(w_1, \cdots, w_d) \in S_w$ are already selected (step S213).

[0117] If it is determined that there exist some combination patterns which are not yet selected, the controller 136 selects a new combination
25 $(w_1, \cdots, w_d) \in S_w$, stores it in the memory 136a (step S214) and causes the processings at and subsequent to step S204 to be executed for this new combination and n . On the other hand, if it is determined at step S213 that all

of combination patterns have been selected, the controller 136 determines whether or not $n=m$ by referring to n in the memory 136a (step S215). If it is determined that $n=m$ does not apply, the controller 136 updates n in the memory 136a by $n+1$ (step S216), and causes the processings at and

5 subsequent to step S204 to be executed. However, if it is determined that $n=m$, an error termination of the processing results (step S217).

[0118] It is to be noted that the processings which take place at steps S204 ~ 216 are equivalent to re-executing the processings in the hash calculator 1033 and the comparator 134 by changing the value of at least part of n , w_u , and z
10 under the control of the controller 136 when the tag output information $a_{k,i}$ and the calculated value c do not match.

On the other hand, if it is determined at step S210 that the hash value c matches the tag output information $a_{k,i}$, the read/write section 135 selects tag ID information id_n which is related to the combination of a plurality of
15 initial elements $f_{u,0}$ corresponding to the hash value c from the database memory 1031, extracts this tag ID information id_n and its corresponding data $data_n$ and sends them to the communication section 132 under the control of the controller 135. In addition, the read/write section 135 receives products distribution information pd from the memory 136a, and writes this products
20 distribution information pd as data $data_n$ which corresponds to the tag ID information id_n into the database memory 1031 as an addition (step S218). The tag ID information id_n and data $data_n$ which are sent to the communication section 132 are transmitted to the reader 120 through the network 140 (step S219).

25 [0119] <features of embodiment 10>
<impossibility of tracing>

In the tag device 1010 in the present embodiment, a hash value of a

bit combination value of elements e_{u, v_u} and the manifold values z is used as tag output information $a_{k, i}$. Accordingly, the output value can be changed by changing the manifold value z without updating the elements e_{u, v_u} . On account of one-way nature of the hash function, a correlation with the output value which is changed in this manner cannot be obtained. In addition, because the manifold value z assumes t kinds of values, it is possible for the tag device to perform t times at maximum of communication which it is difficult to trace without updating elements e_{u, v_u} .

[0120] [efficiency]

10 In the tag device 1010 of the present embodiment, elements e_{u, v_u} in the confidential value memory 11 are updated only once for t times of communications. Accordingly, the amount of the calculations required for updating processing in the tag device 1010 can be reduced by a factor of $1/t$.

Also, a comparison between the hash value c and the tag output information $a_{k, i}$ which takes place in the backend apparatus 1030 can take place T times at maximum without changing a combination of hash values $H^{wu}(f_{u, 0})$. Accordingly, if a permissible number of communications of the tag device 210 (a maximum value of accesses from the reader 120 to the tag device 1010) were increased, the hash processing in the backend apparatus 1030 does not increase significantly.

[0121] [Embodiment 11]

25 An embodiment 11 is a modification of the embodiment 10, and differs from the embodiment 10 in that a manifold value z_u which assumes t_u kinds ($t_u \geq 2$) of values is stored for each u ($u \in \{1, \dots, d\}$) in a manifold value memory of a tag device, and a tag output information $a_{k, i} = G(e_{1, v_1} \mid z_1 \mid \dots \mid e_{d, v_d} \mid z_d)$ for a bit combination value of each element e_{u, v_u} extracted from a confidential value memory and either one of

manifold values z_u is used as an output value. In addition, while each element e_{u, v_u} in the confidential value memory which corresponds to each u ($u \in \{1, \dots, d\}$) is performed once for commutations of t times, in the embodiment 11, the point in time of communication when element e_{u, v_u} is updated is shifted, so that either one of elements $e_{u', v_{u'}}$ ($u' \in \{1, \dots, d\}$) in the confidential value memory is updated each time the tag device delivers the tag output information $a_{k, i}$. This prevents the tag device from being traced if the tag device is tampered with at any point in time of communication.

[0122] Only a distinction over the embodiment 1 and the embodiment 10 will be described below while omitting a description for what is common with the embodiment 1 and the embodiment 10;

Fig. 24 is an illustration of an overall arrangement of an automatic tag identification system 1100 of an embodiment 11. Fig. 25 is a flow chart for describing processing by a tag device 1110, and Fig. 26 is a flow chart for describing part of processing by a backend apparatus 1130. It is to be noted that in Fig. 24, functional arrangements which are common to the embodiment 1 and 8 are designated by like characters as used in Figs. 2 and 15. In addition, while only one tag device 1110 is shown in Fig. 24, in actuality, there are a plurality of tag devices 1110.

[0123] A functional arrangement and a processing method of the present embodiment will be described below with reference to these Figures.

<pre-processing>

Differences with respect to the embodiment 10 reside in that a manifold value generator 1115 of a tag device 1110 sets up a manifold value z_u which assumes t_u kinds ($t \geq 2$) of values for each u ($u \in \{1, \dots, d\}$) and it is stored in a manifold value memory 1116 (equivalent to “first manifold value memory”) and that a manifold value z_u which assumes t_u kinds ($t \geq 2$) of

values for each u ($u \in \{1, \dots, d\}$) is stored in the database memory 1131 (equivalent to “second manifold value memory”) of a backend apparatus 1130. It is to be noted that manifold values z_u which are stored in the database memory 1131 are same as manifold values z_u which are stored in respective tag devices 1110.

[0124] The manifold value generator 1115 can be illustrated by a counter which counts $z_u = 1 \cdots t_u$ for each u ($u \in \{1, \dots, d\}$), a hash calculator which performs a calculation of $z_u = H(\text{seed}, x_u)$, $x_u \in \{1, \dots, t_u\}$ or a hash calculator which performs a calculation of $z_u = H^x(\text{seed})$, $x_u \in \{1, \dots, t_u\}$. In the

description to follow, manifold value z_u is expressed as $z_u = \pi_u(x_u)$, $x_u \in \{1, \dots, t_u\}$. Preferably, π_u is set up such that for an equal value of u , manifold values $z_u = \pi_u(x_u)$ corresponding to $x_u \in \{1, \dots, t_u\}$ do not coincide.

[0125] In the embodiment 11, each x_u is expressed as $x_u = i + \varepsilon_u$ ($u \in \{1, \dots, d\}$) where i represents the number of times a communication is

made by the tag device 1110, and ε_u represents a constant representing an offset from i of each x_u (an integer which satisfies $0 \leq \varepsilon_u \leq r_{\max}$), and where r_{\max} represents a maximum number of times an access is made from the reader 120 to the tag device 1110.

In addition, in the embodiment 11, ε_u and t_u are set up such that at any point in time when a communication is made, $x_u = t_u$ is always satisfied by either one of x_u . For example, t_u corresponding to each u ($u \in \{1, \dots, d\}$) always assumes an equal value while each ε_u is set up such that a set of ε_u ($u \in \{1, \dots, d\}$) becomes a universal set of natural numbers less than t_u .

[0126] It is not necessary that the generation and the storage of manifold values z_u be performed during the pre-processing, but may be performed at the time a communication with the tag device 1110 is processed or at the time a retrieval by the backend apparatus 1130 is processed.

<processing by tag device>

A processing which takes place when the tag device 1110 is read by the reader 120 during an i -th run (i being a natural number) will be described below. It is to be noted that the count x_u ($u \in \{1, \dots, d\}$) has an initial value
5 (at $i=1$) which is equal to $1+\varepsilon_u$, and each count x_u is saved in a memory 115a under the control of the controller 115.

[0127] Initially, a hash calculator 1112 (equivalent to “second calculator”) extracts each element e_{u, v_u} from a confidential value memory 1111 and extracts either manifold value z_u (which is $z_u=\pi_u(x_u)$ in this example) from a
10 manifold value memory 1116 (step S231). The hash calculator 1112 calculates tag output information $a_{k, i} = G(e_{1, v_1} \mid z_1 \mid \dots \mid e_{d, v_d} \mid z_d)$ which is a hash value of a bit combination value (confidential value $s_{k, i}$) of each extracted element e_{u, v_u} and either manifold value z_u (step S232). Where π_u is set up such that manifold values $z_u=\pi_u(x_u)$ corresponding to $x_u \in \{1, \dots, t_u\}$ do not
15 coincide for an equal value of u , it follows that as long as elements in the confidential value memory 1111 are not updated, the manifold value z_u which is used by the hash calculator 1112 in generating tag output information $a_{k, i}$ varies from communication to communication. In addition, a bit combination sequence in the confidential value $s_{k, i} = e_{1, v_1} \mid z_1 \mid \dots \mid e_{d, v_d} \mid z_d$ is not
20 limited thereto. The generated tag output information $a_{k, i}$ is sent to the interface 114, which transmit this tag output information $a_{k, i}$ (step S233).

[0128] Subsequently, in the controller 136, an arithmetic operation of $x_u \leftarrow x_u + 1$ ($u \in \{1, \dots, d\}$) takes place for x_u in the memory 136a (step S234). It is to be noted that in the embodiment 11, ε_u and t_u are set up such that $x_u=t_u$
25 applies for either x_u at any point in time of any communication. Accordingly, as a result of the arithmetic operation $x_u \leftarrow x_u + 1$, $x_u > t_u$ applies for either x_u . The controller 136 then substitutes 1 for x_u which satisfies $x_u > t_u$ (step S235).

In the present embodiment, u which corresponds to this x_u is represented by u' .

Subsequently, the hash calculator 813 extracts a partial element $e_{u', vu'}$ (an element corresponding to above mentioned $u' \in \{1, \dots, d\}$) from the confidential value memory 1111, and calculates a hash value $H(e_{u', vu'})$ of the extracted element $e_{u', vu'}$ (step S236). The hash calculator 813 overwrites the hash value $H(e_{u', vu'})$ as a new element $e_{u', vu'+1}$ into the confidential value memory 1111 (step S237), then completing a processing by the tag device 1110.

[0129] By the processing mentioned above, it follows that each time the interface 114 delivers tag output information $a_{k, i}$, the hash calculator 813 extracts at least one of elements $e_{u', vu'}$ ($u' \in \{1, \dots, d\}$) from the confidential value memory 1111, calculates the hash value $H(e_{u', vu'})$ of the extracted element $e_{u', vu'}$, thus updating the confidential value memory 1111.

<processing by reader>

This remains to be similar to the first mode.

<processing by backend apparatus>

A distinction of the embodiment 11 over the embodiment 10 resides in that the processing indicated at step S26 is performed in place of

processings at steps S208 ~ S213 shown in Fig. 23.

[0130] Specifically, subsequent to the processing at the step S207, the controller 136 selects a combination of $(x_1, \dots, x_d) \in S_x$ as indicated below, and stores it in the memory 136a (step S241).

$$(x_1, \dots, x_d) \in S_x = \{x_1, \dots, x_d \mid x_u \in [0, t_u]\}$$

The controller 136 refers to the combination of $(x_1, \dots, x_d) \in S_x$ in the memory 136a, extracts d manifold values $z_u = \pi(x_u)$ ($u \in \{1, \dots, d\}$) which correspond to this from the database memory 1131, and send them to a hash

calculator 1133, which then calculates a calculated value c which is a result of applying the hash value G to a bit combination value of the hash value $H^{wu}(f_{u,0})$ and the manifold value z_u (step S242). The calculated value c can be illustrated by $c = G(H^{w1}(f_{l,0}) | z_l | \cdots | H^{wu}(f_{u,0}) | z_u | \cdots | H^{wd}(f_{d,0}) | z_d)$, for example, but the bit disposition sequence of each hash value $H^{wu}(f_{u,0})$ and the manifold value z_u is not limited thereto. However, the sequence or the like is required to correspond to the bit disposition sequence of each element in the hash calculator 1112 of the tag device 1110.

[0131] The comparator 134 then reads tag output information $a_{k,i}$ from the memory 136a, receives the calculated value c from the hash calculator 1133, and compares them against each other to determine whether or not $c = a_{k,i}$ (step S243). In this example, the hash value $c = G(H^{w1}(f_{l,0}) | z_l | \cdots | H^{wu}(f_{u,0}) | z_u | \cdots | H^{wd}(f_{d,0}) | z_d)$ and the tag output information $a_{k,i}$ are compared against each other.

In the event it is determined that these do not match, the controller 136 refers to the memory 136a to determine whether or not all of combination patterns $(x_1, \cdots, x_d) \in S_x$ have been selected (step S244). If it is determined that all of combination patterns $(x_1, \cdots, x_d) \in S_x$ have not been selected, the controller 136 selects a new combination $(x_1, \cdots, x_d) \in S_x$, and stores it in the memory 136a, whereupon it causes the processings at and subsequent to step S242 to be executed. On the other hand, if it is determined at step S244 that all of the combination patterns $(x_1, \cdots, x_d) \in S_x$ have been selected, the operation proceeds to step S213 in Fig. 23. On the other hand, if the processing at step S243 reveals a determination that $c = a_{k,i}$, the operation proceeds to step S218 in Fig. 23.

[0132] [efficiency]

The comparison between the hash value c and the tag output

information $a_{k,i}$ in the backend apparatus 1130 takes place $t_1+t_2+\dots+t_{d-1}+t_d$ times at maximum without changing the combination of hash values $H^{wu}(f_{u,0})$. Accordingly, if a permissible number of communications of the tag device 1110 (a maximum value of accesses from the reader 120 to the tag device 1110) increases, the amount of the processing in the backend apparatus 130 does not increase significantly.

[impossibility of tracing]

In the tag device 1110 of the embodiment 11, each time the tag output information $a_{k,i}$ is delivered, either element $e_{u',vu'}$ ($u' \in \{1, \dots, d\}$) which is stored in the confidential value memory 1111 is updated by a hash chain. Accordingly, if the tag device 1110 is tampered with and an element $e_{u',vu'}$ within the confidential value memory 1111 is leaked to an attacker, the attacker cannot find a correlation between the element $e_{u',vu'-t}$ before the updating and an element $e_{u',vu'}$ after updating due to one-way nature of the hash function. Hence, the attacker cannot obtain a correlation between an element acquired from the confidential value memory 1111 and an output value which is delivered from the tag device in the past. In this manner, a tracing of the tag device 1110 can be prevented.

[0133] In addition, in the embodiment 11, if the tag device 1110 is tampered with to leak each manifold value z_u , any element $e_{u',vu'}$ which is stored in the confidential value memory 1111 is updated by overwriting. In this manner, an influence of the tag device 1110 being tampered with can be minimized.

It is to be noted that in the embodiment 11, ε_u and t_u are set up so that it is assured that $x_u=t_u$ is satisfied by either x_u at any point in time of communication. By way of example, a choice is made such that $t_1 = t_2 = \dots = t_d$, and a counter x_u corresponding to each element $e_{u,vu}$ is displaced by 1 ($x_u=i+u/d$).

[0134] However, rather than choosing an equal value for t_u ($u \in \{1, \dots, d\}$), each counter x_u corresponding to each element e_{u, v_u} may be displaced by a spacing which is determined by dividing t_u having a largest value into d sections. In this instance, the requirement of perfectly forward secure may not
5 be satisfied, but it is at least possible to suppress the influence of tampering.

[Embodiment 12]

An embodiment 12 is a modification of the embodiment 11. In a similar manner as in the embodiment 11, in the embodiment 12 also, a point in time of communication when each element e_{u, v_u} is updated is shifted.
10 However, in the embodiment 12, each time a tag device delivers the tag output information $a_{k, i}$ $\sum_{u=1}^d t_u$ times, either element $e_{u', v_{u'}}$ is extracted, and a hash value $H(e_{u', v_{u'}})$ of the extracted element $e_{u', v_{u'}}$ is calculated.

[0135] Specifically, in response to each external access, the tag device of the embodiment 12 counts up a counter $x_u \in \{1, \dots, t_u\}$ corresponding to
15 either one of d elements e_{u, v_u} (for example, counts up by 1 in the sequence of $e_{1, v_1} \dots e_{d, v_d}$). Because the counter x_u corresponds to the manifold value z_u which constitutes the tag output information $a_{k, i} = G(e_{1, v_1} \mid z_1 \mid \dots \mid e_{d, v_d} \mid z_d)$, this tag device is capable of delivering the tag output

information $a_{k, i}$ having different values $\sum_{u=1}^d t_u$ times without updating each
20 element e_{u, v_u} . In the present embodiment, either element e_{u, v_u} is updated each time the tag output information $a_{k, i}$ is delivered $\sum_{u=1}^d t_u$ times. In this manner, the amount of calculations for updating the tag device is minimized while maintaining the manifoldness of output values of the tag device.

[0136] In the following, only a distinction over the embodiments 1 and 11

will be described while omitting a description for what is common with the embodiments 1 and 11.

Fig. 27 is a flow chart for describing processing by the tag device of the embodiment 12. It should be noted that the entire functional arrangement
5 is similar to embodiment 11 (Fig. 24).

A processing method of the present embodiment will be described below with reference to these Figures.

<pre-processing>

In the embodiment 11, a choice is made that $x_u = i + \varepsilon_u$ ($u \in \{1, \dots, d\}$)
10 and ε_u and t_u are set up such that it is assured that $x_u = t_u$ is satisfied by either x_u at any point in time of communication. In the embodiment 12, these limitations are not employed.

[0137] <processing by tag device>

A processing which occurs when a tag device 310 is read by a reader
15 20 during an i -th run (i being a natural number) will be described below. It is to be noted that the count x_u ($u \in \{1, \dots, d\}$) has an initial value (at $i=1$) equal to 1, and u' and u'' also have initial values of 1. It is to be noted that u' corresponds to an element $e_{u', v_{u'}}$ being updated while u'' corresponds to the count $x_{u''}$ of an element $e_{u'', v_{u''}}$ which is being counted up. Each parameter is
20 stored in a memory 136a under the control of the controller 136.

[0138] Initially, the hash calculator 1112 extracts each element e_{u, v_u} from the confidential value memory 1111, and extracts either manifold value z_u (which is $z_u = \pi_u(x_u)$ in this example) from the manifold value memory 1116 (step S241). The hash calculator 1112 then calculates tag output information
25 $a_{k,i} = G(e_{1, v_1} \parallel z_1 \parallel \dots \parallel e_{d, v_d} \parallel z_d)$ which is a hash value of a bit combination value of each extracted element e_{u, v_u} and either manifold value z_u (step S242).

The generated tag output information $a_{k,i}$ is sent to the interface 114,

which transmits the tag output information $a_{k,i}$ (step S243).

[0139] Subsequently, an arithmetic operation of $x_{u''} \leftarrow x_{u''} + 1$ ($u'' \in \{1, \dots, d\}$) is applied to $x_{u''}$ in the memory by the controller 136 (step S244), thus determining whether or not $x_{u''} > t_{u''}$ (where $t_{u''}$ represents a maximum value of $x_{u''}$) (step S245). If it is determined that $x_{u''} > t_{u''}$ does not apply, processings by the tag device 1110 are terminated.

On the other hand, if it is determined that $x_{u''} > t_{u''}$, the controller 136 substitutes $u'' + 1$ for u'' in the memory 136a (step S246), and determines whether or not $u'' > d$ (step S247). If $u'' > d$ does not apply, the processings by the tag device 1110 are terminated, but if $u'' > d$ applies, the hash calculator 813 extracts an element $e_{u', v_{u'}}$ (an element corresponding to $u' \in \{1, \dots, d\}$ mentioned above) from the confidential value memory 1111, and calculates a hash value $H(e_{u', v_{u'}})$ of the extracted element $e_{u', v_{u'}}$ (step S248). The hash calculator 813 then saves this hash value $H(e_{u', v_{u'}})$ as a new element $e_{u', v_{u'}+1}$ in the confidential value memory 1111 by overwriting (step S249).

[0140] Subsequently, an arithmetic operation of $v_{u'} \leftarrow v_{u'} + 1$ (a number of updates) takes place in the hash calculator 813, for example, (step S250) and it is determined whether or not $v_{u'}$ has exceeded a maximum value (max) of the number of updates for the element $e_{u', v_{u'}}$ (step S251). If it is determined that $v_{u'} > \text{max}$ does not apply, processings by the tag device 1110 are terminated, but if $v_{u'} > \text{max}$, arithmetic operations of $u' \leftarrow u' + 1$ (a change of the element being updated) and $v_{u'} \leftarrow 0$ (resetting of the number of updates for the element being updated) take place in the controller 136 (step S252), and these results are stored in the memory 136a before terminating the processing by the tag device 1110.

[0141] By the processings mentioned above, it is seen that each time the

interface 114 delivers tag output information $a_{k,i}$ $\sum_{u=1}^d t_u$ times, the hash calculator 813 extracts either element $e_{u',vu'}$ from the confidential value memory 1111 and calculates a hash value $H(e_{u',vu'})$ of the extracted element $e_{u',vu'}$ to update the confidential value memory 11.

5 <processing by reader> }

This remains to be similar to the embodiment 8.

<processing by backend apparatus>

This remains to be similar to the embodiment 11.

[0142] <features of embodiment 12>

10 [efficiency]

In the embodiment 12, either element $e_{u',vu'}$ is updated each time the

tag device 1110 performs a communication $\sum_{u=1}^d t_u$ times, and accordingly an

amount of calculations for updating processing in the tag device 1110 can be reduced. In other words, in the present embodiment, tag output information

15 $a_{k,i} = G(e_{1,v1} \mid z_1 \mid \cdots \mid e_{d,vd} \mid z_d)$ is generated and delivered while

substituting manifold values which are equal in number to $\sum_{u=1}^d t_u$ for every

communication. Accordingly, during the communications which are equal in

number to $\sum_{u=1}^d t_u$, the manifoldness of the output value of the tag device can

be secured without updating the element $e_{u,vu}$. By updating either element $e_{u',$

20 vu' for every $\sum_{u=1}^d t_u$ communications, the manifoldness of the output during

the following $\sum_{u=1}^d t_u$ communications can be secured. Since the element $e_{u',vu'}$

is updated only once for $\sum_{u=1}^d t_u$ communications, the amount of calculations for updating in the tag device 1110 can be minimized.

[0143] [impossibility of tracing]

The tag device 1110 of the present embodiment is constructed such

5 that each time the interface 114 delivers tag output information $a_{k,i}$ $\sum_{u=1}^d t_u$

times, the hash calculator 813 updates the confidential value memory 1111.

Accordingly, if the tag device 1110 is tampered with to leak elements $e_{u'}$, $v_{u'}$ in the confidential value memory 1111 to an attacker, the number of past output values from the tag device 1110 which the attacker can know is less than

10 $\sum_{u=1}^d t_u$. This allows the tracing of the tag device 110 to be suppressed while reducing the amount of processing for update calculations of the tag device 1110.

[0144] [Embodiment 13]

An embodiment 13 is a modification of the embodiments 1 to 4, and
15 6 to 12, and is characterized in two kinds of hash functions $G(x)$ and $H(x)$ used. In the description to follow, only these hash functions $H(x)$ and $G(x)$ will be described.

<No. 1>

The hash function $G(x)$ in this example is $\text{hash}(1 \mid x)$ where hash
20 represents a hash function for $\{0, 1\}^* \rightarrow \{0, 1\}^r$ where r represents a natural number, and the hash function $H(x)$ is $\text{hash}(0 \mid x)$. It should be noted that $\alpha \mid \beta$ represents a bit combination of α and β . The hash function $G(x)$ may be chosen to be $\text{hash}(0 \mid x)$ while the hash function $H(x)$ may be chosen to be $\text{hash}(1 \mid x)$.

[0145] <No. 2>

The hash function $H(x)$ (first function F1) in this example is $\text{hash}(p \mid x)$ where r, s are natural numbers, hash represents a hash function for $\{0, 1\}^* \rightarrow \{0, 1\}^r$, and $p \in \{0, 1\}^s$. The hash function $G(x)$ (second function F2) is
5 $\text{hash}(q \mid x)$ where $q \in \{0, 1\}^s$ and $p \neq q$.

<No. 3>

The hash function $H(x)$ (first function F1) is $\text{hash}(\text{pad}(x, p))$ when p padded to x (a padding of p to x) is represented by $\text{pad}(x, p)$ where $p \in \{0, 1\}^s$. The function $G(x)$ (second function F2) is $\text{hash}(\text{pad}(x, q))$ when q padded to x
10 (a padding of q to x) is represented as $\text{pad}(x, q)$ where $q \in \{0, 1\}^s$ and $p \neq q$. A padding position (a position in a bit train) of p or q relative to x is not restricted in particular. By way of example, p or q may be bit combined before or after x or may be inserted into a bit train of x .

[0146] <No. 4>

15 The hash function $H(x)$ (first function F1) is $\text{hash}(x)$ where hash represents a hash function for $\{0, 1\}^* \rightarrow \{0, 1\}^r$, and the function $G(x)$ (second function F2) is $\text{hash}(rx)$ where rx represents a bit inversion of x .

<effect of embodiment 13>

In the present embodiment, while using only one kind of hash
20 function, two kinds of hash calculations $G(x)$, $H(x)$ can be implemented without collapsing the properties thereof (one-way nature, delivering random values). This allows a circuit scale which constitutes a hash function to be reduced, with a consequence that the scale of a circuit which is assembled into a tag device can be made smaller, realizing a reduction in the cost of a tag
25 device.

[0147] [Second Mode]

<arrangement>

A second mode for carrying out the present invention will now be described.

In this mode, an updater which is provided externally of a tag device updates a privileged ID information which is stored in a tag device to a new
5 privileged ID information, the association of which with respect to the privileged ID information is difficult to follow at a given opportunity.

<arrangement>

Fig. 28 is a block diagram illustrating a schematic arrangement of the present mode.

10 [0148] An illustrated in Fig. 28, an updating system 1500 of the present mode comprises a tag device 1510 and a security server 1560 which is provided externally thereof.

A tag device 1510 comprises a confidential value memory which stores privileged ID information which has privileged a tag ID information
15 which is inherent to each tag device, a read/write section 1512 electrically connected with the confidential value memory, a first output section 1513 and a second input section 1514 electrically connected with the read/write section 1512.

A security server 1560 comprises a first input section 1561, an
20 updater 1562 electrically connected with the first input section 1561, and a second output section connected to the updater 1562.

[0149] <update processing of privileged ID>

An updating of a privileged ID takes place as described below.

Initially, at a given opportunity, the read/write section 1512 of the tag
25 device 1510 reads out privileged ID information sid_h which is stored in the confidential value memory 1511, and the first output section 1513 delivers the privileged ID information sid_h to the security server 1560 which is provided

externally of respective tag devices.

The first input section 1561 of the security server 1560 accepts an input of privileged ID information sid_h . And the updater 1562 generates new privileged ID information sid_h' , the association of which with the privileged ID information sid_h is difficult to follow, and the second output section 1563 delivers the new privileged ID information sid_h' to the tag device 1510.

[0150] The second input section 1514 of the tag device 1510 accepts an input of the new privileged ID information sid_h' , and the read/write section 1512 stores the new privileged ID information sid_h' in the confidential value memory 1511.

[Embodiment 14]

Fig. 29 is a conceptual view illustrating an overall arrangement of an updating system 2000 of an embodiment 14.

As illustrated in this Figure, the updating system 2000 comprises a tag device 2010 such as a radio tag which is applied to goods, a client apparatus 2020, a backend apparatus 2050 which controls products distribution information or the like which relates to ID in plain text, and a security server 2060 which performs a restoration of ID, re-privileging of privileged ID (a server which re-privileges the privileged ID which is transmitted through the network and is equivalent to “updater”). The client apparatus 2020, the backend apparatus 2050 and the security server 2060 are connected to be capable of communication with each other through a network 2070 such as internet or the like. It is to be noted that the client apparatus 2020 has the function to operate as a reader described above in connection with the first mode. The effect which is implemented with the tag device, the reader and the backend apparatus in the first mode is implemented by the tag device 2010, the client apparatus 2020, the backend apparatus 2050 and the

security server 2060. While one tag device 2010, client apparatus 2020, backend apparatus 2050 and security server 2060 are illustrated in this Figure, it should be understood that normally there are a plurality of tag devices, and a plurality of client apparatus, backend apparatus and security servers may also be provided.

[0151] The client apparatus 2020 of this example initially reads the privileged ID from the tag device 2010, and sends it to the security server 2060. The security server 2060 restores ID from the privileged ID, and returns this ID to the client apparatus 2020. Upon receiving the ID, the client apparatus 2020 accesses the backend apparatus 2050 to demand an entry of information such as ID, a date and time of reading, a location of reading, a temperature or the like and an acquisition of information relating to ID or the like. A manner of utilizing a proxy model can be contemplated that the client apparatus 2020 transmits the privileged ID to the security server 2060, which then directly accesses the backend apparatus 2050. What is characterizing the present embodiment is that an apparatus such as the security server 2060 or the like which is provided externally of the tag device 2010 re-privileges the privileged ID within the tag device 2010 (namely, updates the privileged ID to a separate privileged ID).

[0152] Fig. 30 is a block diagram illustrating a functional arrangement of an updating system 1 of the present embodiment.

<tag device>

A tag device 2010 in this example comprises a confidential value memory 2011, a read/write section 2012 (equivalent to “first read/write section”), an interface 2013 (equivalent to “first output section” and “second input section”), a memory 2014a and a controller 2014.

Here, each of the confidential value memory 2011 and the memory

2014a is an RAM (Random Access Memory) capable of read/write operation such as EEPROM (Electrically Erasable and Programmable Read Only Memory), FeRAM (Ferroelectric Random Access Memory), a flash memory, NV (Nonvolatile) RAM or the like, for example. The read/write section 2012
5 is a hardware which reads/writes data from or into the confidential value memory 2011 at a given address under the control of the controller 2014. The controller is an integrated circuit constructed to control processing by the entire tag device 2010, for example.

[0153] The interface 2013 is a hardware which inputs or outputs data with
10 respect to the client apparatus 2020 by the radio or wire communication.

Specifically, the interface 2013 comprises an encoding/decoding circuit which performs an encoding/decoding according to NRZ code, a Manchester code, Miller code, a single polarity RZ code or the like, a modulator/demodulator which performs a modulation/demodulation in accordance with ASK

15 (Amplitude Shift Keying), PSK (Phase Shift Keying), FSK (Frequency Shift Keying) or the like, and an antenna such as a dipole antenna, a microstrip antenna, a loop coil, a cored coil or the like for performing a signal transmission and reception using a frequency in a low frequency band or ISM band (Industry Science Medical band). A communication system used may
20 comprise an electromagnetic induction system or a radio wave system, for example.

[0154] The confidential value memory 2011 is electrically connected with the read/write section 2012, which is in turn electrically connected with the interface 2013. While not shown in this Figure, the controller 2014 is

25 electrically connected to various parts of the tag device 2010.

<client apparatus>

The client apparatus 2020 of this example comprises an interface

2022, a communication section 2021, a memory 2024a and a controller 2024.

A products distribution information memory 121 is a magnetic recorder such as a hard disc unit, a flexible unit or the like, an optical disc unit such as DVD-RAM (Random Access Memory), CD-R (Recordable) / RW (ReWritable) or the like, a magneto-optical such as MO (Magneto-Optical disc) or the like, or a semiconductor memory such as EEPROM (Electronically Erasable and Programmable Read Only Memory), a flash memory or the like. The interface 2022 is a hardware which is similar to the example of the interface 2013. The communication section 2021 comprises LAN card, a modem, a terminal adapter or the like, and the controller 2023 is CPU (Central Processing Unit) of CISC (Complex Instruction Set Computer) type, RISC (Reduced Instruction Set Computer) type, or the like and including a memory 2023a.

[0155] The interface 22 and the communication section 2021 are electrically connected together, and while omitted from illustration in this Figure, the controller 2024 is electrically connected to various parts of the client apparatus 2020.

<updater>

The security server 2060 comprises a communication section 2062 (equivalent to “first input section” and “second output section”), a random number generator 2063, a read/write section 2064 (equivalent to “second read/write section”), a privileged ID memory 2061, a memory 2065a and a controller 2065. It is to be noted that the random number generator 2063, the read/write section 2064 and the privileged ID memory 2061 constitute together “updating section”. Specifically, the security server 2060 is constituted by a known computer of Neumann type which is formed by connecting together a CPU, an RAM, an ROM (Read Only Memory), an

external memory such as a magnetic recorder or an optical disc unit, a LAN card, a modem, a terminal adapter and like through busses, for example, and which executes a given program. Each of processing functions indicated below are implemented by CPU which reads a program stored in the RAM
5 and executes a processing in accordance therewith.

[0156] <processing>

Fig. 31 is a flow chart for describing a processing procedure of the present embodiment. A functional arrangement and a processing of the present embodiment will be described below with reference to Figs. 29 to 31.

10 It is to be understood that the tag device 2010, the client apparatus 2020 and the security server 2060 execute respective processings under the control of the controllers 2014, 2023 and 2065, respectively. Data which are processed are sequentially stored in the memory 2014a, 2023a or 2065a, which is accessed as a processing such as a calculation takes place, but a description
15 for this will be omitted in the description to follow.

[0157] <pre-processing>

Privileged ID information in this example is a random value r_h which is related to tag ID information id_h . A random value r_h corresponding to the tag ID information id_h which is inherent to the tag device 2010 is stored as
20 privileged ID information sid_h in the confidential value memory 2011 of the tag device 2010. Tag ID information id_1, \dots, id_m which corresponds to each tag device 2010 and privileged ID information or random values r_1, \dots, r_m which are related to the respective the tag ID information are stored in a privileged ID memory 2061 of the security server 2060. It is to be noted that h
25 represents a natural number equal to or greater than 1 and equal to or less than m , and represents the number of each tag device 2010. m presents a total number of tag devices.

<privileged ID information updating processing>

Initially, utilizing some sort of authentication technology, an authentication of each other is made between the client apparatus 2020 and security server 2060. A communication between the client apparatus 2020 and security server 2060 takes place under encryption according to some sort of encryption technology.

[0158] An updating processing of the privileged ID information sid_h is initiated by a trigger, which may be a passage through a location such as a porch which a man never fails to pass when he leaves a house, a number of times the privileged ID information stored in the tag device 2010 is used (a count reaching a given value) or the like. In response to the trigger, the client apparatus 2020 initially transmits a read command to the tag device 2010 from its interface 2022 (step S301). This read command is received by the interface 2013 of the tag device 2010, and this triggers the read/write section 2012 to extract the privileged ID information sid_h from the confidential value memory 2011 (step S302). The extracted privileged ID information sid_h is transmitted (delivered) from the interface 2013 to the client apparatus 2020 (step S303). The privileged ID information sid_h is received by the interface 2022 of the client apparatus 2020, and is transmitted together with a solicitation to update the privileged ID information (a demand to re-privilege) by the communication section 2021 to the security server 2060 through the network 2070 (step 304).

[0159] Information such as the privileged ID information (sid_h) or the like is received by the communication section 62 of the security server 2060 (its input is accepted) (step S305), and is sent to a read/write section 2064. This also triggers the random number generator 63 (equivalent to “random value generator”) to generate a random number r_h' (step S306).

The generation of the random number r_h' is carried out so as to avoid assuming a same value as privileged ID information in the privileged ID memory 2061. This generation takes place by using a pseudo-random number generating algorithm which is based on a calculation theory constructed with a one-way hash function such as SHA-1, for example, or the like, and the generated random number r_h' is sent to the read/write section 2064. The read/write section 2064 retrieves (selects) tag ID information id_h corresponding to the privileged ID information sid_h from the privileged ID memory 2061, relates the random number r_h' (equivalent to “random value”) as a new privileged ID information sid_h' to the tag ID information id_h , and stores it in the privileged ID memory 2061 (step S307). In addition, the read/write section 2064 sends the new privileged ID information $sid_h' = r_h'$ to a communication section 2062, which then transmits the new privileged ID information sid_h' to the client apparatus 2020 (equivalent to “delivered to tag device”) through the network 2070 (step S308).

[0160] The transmitted new privileged ID information sid_h' is received by the communication section 2021 of the client apparatus 2020, and is transmitted to the tag device 2010 from the interface 2022 (step S309). The tag device 2010 receives the new privileged ID information sid_h' at the interface 2013 (accepts an input) and sends it to the read/write section 2012, which sends the new privileged ID information sid_h' to the confidential value memory 2011 to be stored therein (step S310). Subsequently, the tag device 2010 sends this new privileged ID information sid_h' to the backend apparatus 2050 through the reader in response to a read demand from the reader (not shown). The backend apparatus 2050 sends the received privileged ID information sid_h' to a database memory 1131, which receives it at its communication section 2062 and sends it to the read/write section 2064. The

read/write section 2064 retrieves a random value which coincides with the privileged ID information sid_h' from the privileged ID memory 2061, reads out the tag ID information id_h which is related to the coinciding random value r_h and sends it to the communication section 2062, which then transmits it to the backend apparatus 2050.

[0161] <features of embodiment 14>

In the present embodiment, the privileged ID information which is stored in the tag device 2010 can be updated at arbitrary timing. This allows a tracing of the tag device 2010 on the basis of a common character of privileged ID information which remains in the history of communications or the like to violate the privacy to be avoided. Since a random value is used as privileged ID information, an attacker cannot know the association between privileged ID information before and after the update. Accordingly, a firm prevention of a tracing of the tag device 2010 can be realized. In addition, since a complicated re-privileging processing is performed in the security server 2060 which is external to the tag device 2010, there is no need to provide a circuit which would be required for purpose of re-privileging processing within the tag device 2010 itself. As a consequence, the cost of the tag device 2010 itself can be suppressed low.

[0162] [Embodiment 15]

This embodiment is a modification of the embodiment 14, and differs from the embodiment 14 in that an encrypted text according to a common key encryption technique is used as privileged ID information. In the following, a distinction over the embodiment 14 will be principally described.

Fig. 32 is a block diagram illustrating a functional arrangement of an updating system 2100 of the present embodiment. Fig. 33 is a flow chart for describing a processing procedure therefor. A functional arrangement and a

processing of the present embodiment will be described below with reference to these Figures. It is to be noted that arrangements which are common to the embodiment 14 are designated in Fig. 32 by like characters as used in Fig. 30. A security server 2160 executes various processings under the control of a controller 2065. A key memory 2161, a read/write section 2064 and an ID extractor 2166, an encryptor 2167 and a random number generator 2063 constitute together “updating section”.

[0163] <pre-processing>

Privileged ID information of this mode represents information including a first encrypted text according to a common key encryption technique such as AES and a key ID information which corresponds to the common key used in this encryption. In this example, privileged ID information of a tag device 2110 is defined as $sid_h = (ek_j(id_h | r), kid_j)$ where h is a natural number equal to or greater than 1 and equal to or less than m , and represents the number corresponding to the tag device 2110 and j is a natural number equal to or greater than 1 and equal to or less than n and represents the number corresponding to each key. m represents a total number of tag devices and n represents a total number of keys. In addition, k_j represents a j -th common key, kid_j key ID information corresponding to the common key k_j and r a random number. In addition, $ek(\alpha)$ represents an encrypted text which is obtained by encrypting α according to the common key encryption technique using a common key k , and $\alpha | \beta$ represents a bit combination of α and β .

[0164] Privileged ID information $sid_h = (ek_j(id_h | r), kid_j)$ which corresponds to tag ID information id_h is stored in the confidential value memory 2111 of the tag device 2110 of this example. Each key ID information (kid_1, \dots, kid_n) and a common key (k_1, \dots, k_n) of the common encryption technology are

stored in a key memory 2161 of a security server 2160 (equivalent to “updater”) in a manner relating to each other. In addition, information relating to the magnitude (bit length) and the padding position of a random number r appearing in $sid_h = (ek_j(id_h | r), kid_j)$ is stored in a memory 2065a.

- 5 [0165] It is to be noted that in this example, it is assumed that the total number m of the tag devices is sufficiently greater than the total number of keys ($m \gg n$) such that identical key ID information is allotted to tag devices which are not related to each other. Specifically, for example, rather than
10 allotting the identical key information to tag devices which are applied to a same variety of goods, the identical key ID information is allotted to tag devices which are applied to unrelated goods. In this manner, it is possible to prevent the variety of goods or a specific one of goods from being identified on the basis of the key ID information.

<privileged ID updating processing>

- 15 In the similar manner as in the embodiment 14, the client apparatus 2020 transmits a read command to the tag device 2110 initially (step S320). The tag device 2110 extracts privileged ID information ($sid_h = (ek_j(id_h | r), kid_j)$) from the confidential value memory 2111 (step S321), and transmits it to the client apparatus 2020 (step S322). In response thereto, the client
20 apparatus 320 transmits the privileged ID information sid_h together with a solicitation for update to the security server 2160 (step 323).

- [0166] Information including the privileged ID information sid_h is received by the communication section 2062 of the security server 2160 (step S324), and the first encrypted text $ek_j(id_h | r)$ which constitutes the privileged ID
25 information sid_h is sent to an ID extractor 2166 while the key ID information kid_j is sent to the read/write section 2064. kid_j is also recorded in the memory 2065a.

Upon receiving the key ID information kid_j , the read/write section 2064 extracts the common key k_j which corresponds to the key ID information kid_j from the key memory 2161 and sends it to the ID extractor 2166 (step S325). Upon receiving it, ID extractor 2166 decrypts the first encrypted text ($ek_j(id_h | r)$) using the common key k_j and extracts tag ID information id_h . Specifically, the ID extractor 2166 calculates ($id_h | r$) from $id_h = dk_j(ek_j(id_h | r))$, and extracts id_h using information relating to the magnitude and the padding position of the random number r which is stored in the memory 2065a (step S326). Here, $dk(\alpha)$ represents a decryption of an encrypted text α with the common key k . The calculated tag ID information id_h is sent to an encryptor 2167 together with the common key k_j . The random number generator 2063 generates a random number r' , and sends it to the encryptor 2167 (step S327). Using the common key k_j , the tag ID information id_h and the random number r' which are sent thereto and information relating to the magnitude and the padding position of the random number which is stored in the memory 2065a, the encryptor 2167 generates (calculates) a second encrypted text ($ek_j(id_h | r')$) (a second encrypted text being one, the association of which with the first encrypted text is difficult to follow), and sends it to the communication section 2062 (step S328).

[0167] The communication section 2062 transmits (delivers) the encrypted text ($ek_j(id_h | r')$) which is sent thereto and the key ID information kid_j in the memory 2065a as a new privileged ID information ($sid_h' = (ek_j(id_h | r'), kid_j)$) (step S329).

The new privileged ID information sid_h' which is transmitted is received by the client apparatus 2020 through the network 2070 in the similar manner as in the embodiment 14, and is transmitted to the tag device 2110 (step S330). The tag device 2110 receives the new privileged ID information

sid_h' at its interface 2013, and the read/write section 2012 stores it in the confidential value memory 2111 (step S331), and in response to a subsequent read command from the reader, sends the new privileged ID information sid_h' to the backend apparatus 2050 through the reader. The backend apparatus 2050 sends the received the privileged ID information sid_h' to the security server 2160, which receives it at its communication section 2062. The security server 2160 then decrypts the tag ID information by a procedure which is similar to steps S324 and 325, and sends it through the communication section 2062 and the network 2070 to the backend apparatus 2050.

[0168] <features of embodiment 15>

In the present embodiment, information including an encrypted text which is formed according to the common key encryption technique is used as privileged ID information, and accordingly, an attacker who does not know the common key cannot know an association of the privileged ID information before and after the update. In this manner, a firm prevention of a tracing of the tag device 2010 can be realized.

While an encrypted text formed by an exclusive logical sum of the random number and ID constitutes privileged ID information in the present embodiment, the privileged ID information may be constituted by any other method as long as the property of probability encryption (the property that if a same ID is encrypted with a same key, a different encrypted text can be delivered) is maintained. The same is true in an embodiment 16.

[0169] [Embodiment 16]

An embodiment 16 is a modification of the embodiment 14, and differs from the embodiment 14 in that an encrypted text formed according to a public key encryption technique is used as privileged ID information. A

distinction over the embodiment 14 will be principally described below.

Fig. 34 is a block diagram illustrating a functional arrangement of an updating system 2200 of the present embodiment, and Fig. 35 is a flow chart for describing a processing procedure therefor. A functional arrangement and a processing according to the present embodiment will be described below with reference to these Figures. It is to be noted that in Fig. 34, arrangements which are common to the embodiment 14 are designated by like characters as used in the embodiment 14. A read/write section 2064, a key memory 2261, an ID extractor 2266, an encryptor 2267 and a random number generator 2063 constitute together an “updating section”.

[0170] <pre-processing>

Privileged ID information in this mode comprises information including a first encrypted text according to a public key encryption technique such as RSA and a key ID information which corresponds to the public key used in the encryption. In this example, privileged ID information of a tag device 2210 is defined as $\text{sid}_h = (\text{epk}_j(\text{id}_h \parallel r), \text{kid}_j)$. It is to be understood that pk_j represents a j -th public key, kid_j key ID information which corresponds to the public key pk_j and $\text{epk}(\alpha)$ an encrypted text formed by encrypting α according to the public key encryption technique using the public key pk .

[0171] Privileged ID information $\text{sid}_h = (\text{epk}_j(\text{id}_h \parallel r), \text{kid}_j)$ is stored in a confidential value memory 2211 of the tag device 2210 of this example. Each key ID information $(\text{kid}_1, \dots, \text{kid}_n)$, a common key $(\text{sk}_1, \dots, \text{sk}_n)$ and a public key $(\text{pk}_1, \dots, \text{pk}_n)$ of the public key encryption technique (a key pair $(\text{sk}_j, \text{pk}_j)$) are stored in a manner relating to each other in a key memory 2261 of a security server 2260 (equivalent to “updater”). Information relating to the magnitude (bit length) and the padding position (bit position) of the random

number r in $sid_h=(epk_j(id_h | r), kid_j)$ are stored in a memory 2065a of this example.

[0172] In the similar manner as in the embodiment 15, in this example, an identical key ID information is allotted to unrelated tag devices. This allows
5 the variety of goods or a specific one of goods to be prevented from being identified on the basis of the key ID information.

<privileged ID updating processing>

In the similar manner as in the embodiment 14, a client apparatus 2020 initially transmits a read command to the tag device 2210 (step S340).

10 The tag device 2210 extracts privileged ID information $sid_h=(epk_j(id_h | r), kid_j)$ from the confidential value memory 2211 (step S341), and transmits it to the client apparatus 2020 (step S342). In response thereto, the client apparatus 2020 transmits the privileged ID information sid_h together with a solicitation for update to the security server 2260 (step 343).

15 [0173] Information including the privileged ID information sid_h or the like is received by the communication section 2062 of the security server 2260 (step S344), and the first encrypted text $epk_j(id_h | r)$ which constitutes the privileged ID information sid_h is sent to the ID extractor 266 while the key ID information kid_j is sent to the read/write section 2064. The key ID information
20 kid_j is also recorded in the memory 2065a.

Upon receiving the key ID information kid_j , the read/write section 2064 extracts a secret key sk_j and a public key pk_j (or key pair) which corresponds to the key ID information kid_j from the key memory 2261, and sends the secret key sk_j to the ID extractor 2266 and sends the public key pk_j
25 to the encryptor 2267, respectively (step S345). Upon receiving the secret key sk_j , the ID extractor 2266 decrypts the first encrypted text $epk_j(id_h | r)$ using the secret key sk_j , and extracts tag ID information id_h . Specifically, $(id_h | r)$ is

calculated according to $id_h = dsk_j(epk_j(id_h | r))$, and id_h is calculated using information relating to the magnitude and the padding position of the random number r in the memory 2065a (step S346). Here, $dsk(\alpha)$ represents a decryption of an encrypted text α with the secret key sk . The calculated tag ID
5 information id_h is sent to the encryptor 2267. The random number generator 2063 generates a random number r' , and sends it to the encryptor 2267 (step S347). Using the public key pk_j , the tag ID information id_h and the random number r' which are sent thereto and information relating to the magnitude and the padding position of the random number, the encryptor 2267 generates
10 (calculates) an encrypted text $(epk_j(id_h | r'))$ (a second encrypted text, the association with the first encrypted text of which is difficult to follow), and sends it to the communication section 2062 (step S348).
[0174] The communication section 2062 transmits (delivers) the second encrypted text $(epk_j(id_h | r'))$ which is sent thereto and the key ID information
15 kid_j in the memory 2065a as a new privileged ID information $sid_h' = (epk_j(id_h | r'), kid_j)$ (step S349).

The transmitted new privileged ID information sid_h' is received by the client apparatus 2020 through the network 2070 in the similar manner as in embodiment 14, and is transmitted to the tag device 2210 (step S350). The
20 tag device 2210 causes its read/write section 2012 to store the new privileged ID information sid_h' in the confidential value memory 2211 (step S351). In response to a subsequent read command from the reader, this new privileged ID information sid_h' is sent to the backend apparatus 2050 through the reader. The backend apparatus 2050 sends the received privileged ID information
25 sid_h' to the security server 2260, which then receives it by the communication section 2062. Subsequently, the security server 2260 decrypts tag ID information by a procedure similar to steps S345 and 346, and sends it to the

backend apparatus 2050 through the communication section 2062 and the network 2070.

[0175] <features of embodiment 16>

Since information containing an encrypted text formed according to the public key encryption technique is used as privileged ID information in the present embodiment, an attacker who does not know the secret key cannot know the association between the privileged ID information before and after the update. In this manner, a firm prevention of a tracing of the tag device 2210 can be realized.

10 [Embodiment 17]

This embodiment is a modification of the embodiment 14, and differs from the embodiment 14 in that privileged ID information is updated using an encryption algorithm having the property of re-encryption (the property of an encryption capable of generating a different encrypted text data only using encrypted data and the public key. A decryption takes place by using the same secret key). A distinction over the embodiment 14 will be principally described below.

[0176] Fig. 36 is a block diagram illustrating a functional arrangement of an updating system 2300 of the present embodiment, and Fig. 37 is a flow chart for describing a processing procedure therefor. A functional arrangement and a processing of the present embodiment will be described below with reference to these Figures. It is to be noted that in Fig. 36, arrangements which are common with the embodiment 14 are designated by like characters as used in the embodiment 14. A security server 2360 executes various processings under the control of a controller 2065. A key memory 2361, a read/write section 2064, a random number generator 2063, a remainder multiplication calculator 2366 and a remainder power calculator 2367

constitute together an “updating section”.

[0177] <pre-processing>

Privileged ID information according to this mode is information including a first encrypted text according an encryption algorithm (public key encryption technique) which has the property of re-encryption, and key ID information which corresponds to the public key used in the encryption. In this example, ElGamal encryption (see Tatsuaki Okamoto and Hirosuke Yamamoto, “Modern encryption”, 1998, p118 ~ 119, for example) is used, and privileged ID information of a tag device 2310 is defined as

10 $sid_h = (g^r \bmod p, id_h \cdot pk_j^r \bmod p, kid_j)$. Here, g represents a publicized element of generation, p a sufficiently large primary number, r an arbitrary integer equal to or greater than 0 and equal to or less than $p-1$, $pk_j = g^{x_j} \bmod p$ a j -th public key, sk_j a j -th secret key, and $(g^r \bmod p, id_h \cdot pk_j^r \bmod p)$ an encrypted text. It is to be noted that super-index “ sk_j ” in $pk_j = g^{sk_j} \bmod p$ means
15 “ sk_j ”. In the description to follow and associated Figures, “ $\bmod p$ ” will be omitted from description.

[0178] Privileged ID information $sid_h = (g^r, id_h \cdot pk_j^r, kid_j)$ is stored in a confidential value memory 2311 of the tag device 2310 of the present example. In addition, each key ID information (kid_1, \dots, kid_n) and a public
20 key (pk_1, \dots, pk_n) are stored in a key memory 2361 of a security server 2360 (equivalent to “updater”) in a manner relating to each other. An element of generation g is stored in a memory 2065a.

Again in this example, an identical key ID information is allotted to unrelated tag devices. This allows an identification of the variety of goods or
25 a particular one of goods on the basis of the key ID information to be prevented.

[0179] <privileged ID updating processing>

In the similar manner as in the embodiment 14, a client apparatus 2020 initially transmits a read command to the tag device 2310 (step S360).

The tag device 2310 extracts privileged ID information

$sid_h = (g^r, id_h \cdot pk_j^r, kid_j)$ from the confidential value memory 2311 (step

5 S361), and transmits it to the client apparatus 2020 (step S362). In response thereto, the client apparatus 2020 transmits the privileged ID information sid_h together with a solicitation to update to the security server 2260 (step S363).

[0180] Information inclusive of the privileged ID information sid_h is

received by the communication section 2062 of the security server 2360 (step
10 S364), and $(g^r, id_h \cdot pk_j^r)$ which constitutes the privileged ID information sid_h is sent to a remainder multiplication calculator 2366 (which constitutes “encrypting section”) while kid_j is sent to the read/write section 2064. kid_j is also recorded in the memory 2065a.

Upon receiving key ID information kid_j , the read/write section 2064
15 extracts a public key pk_j which corresponds to this key ID information kid_j from the key memory 2361, and sends it to a remainder power calculator 2367 (which constitutes “encrypting section”) (step S365). This triggers the random number generator 2063 to generate a random number r' which is equal to or greater than 0 and equal to or less than $p-1$, and to send it to the remainder
20 power calculator 2367 (step S366). The remainder power calculator 2367 calculates $(g^{r'}, pk_j^{r'})$ using the element of generation g in the memory 2065a, the public key pk_j and the random number r' which are received, and sends a result of calculation to the remainder multiplication calculator 2366 (step S367). The remainder multiplication calculator 2366 calculates $(g^{r+r'},$
25 $id_h \cdot pk_j^{r+r'})$ using $(g^{r'}, pk_j^{r'})$ and $(g^r, id_h \cdot pk_j^r)$ which are received, and sends a result of calculation as an encrypted text (a second encrypted text) to the communication section 2062 (step S368).

[0181] The communication section 2062 transmits the sent encrypted text $(g^{r+r'}, id_h \cdot pk_j^{r+r'})$ (a second encrypted text, the association with the first encrypted text of which is difficult to follow) and key ID information kid_j in the memory 2065a as new privileged ID information ($sid'_h = (g^{r+r'}$
5 $, id_h \cdot pk_j^{r+r'}, kid_j)$) (step S369).

The transmitted new privileged ID information sid'_h is received through the network 2070 by the client apparatus 2020 in the similar manner as in the embodiment 14, and is transmitted to the tag device 2310 (step S370). The read/write section 2012 of the tag device 2310 then stores the new
10 privileged ID information sid'_h in the confidential value memory 2311 (step S371). Subsequently, the tag device 2310 responds with this new privileged ID information (sid'_h) in response to a subsequent read command.

[0182] <features of embodiment 17>

In this embodiment, privileged ID information is updated using an
15 encryption algorithm having a property of re-encryption, and accordingly, the privileged ID information can be updated without decrypting ID in plain text. Consequently, there is no likelihood that ID be eavesdropped during the updating processing of privileged ID information and a firm prevention of a tracing of the tag device 2310 can be realized.

20 While a public key (pk_1, \dots, pk_n) is stored in the key memory 2361 of the security server 2360 in the present embodiment, an arrangement can be used in which the security server 2360 does not maintain the public key (pk_1, \dots, pk_n) , but acquires the public key (pk_1, \dots, pk_n) from a given public key server for use.

25 [0183] While ElGamal encryption is used in the present embodiment, a different algorithm such as a higher order remainder encryption may also be used provided it is an encryption algorithm having the property of

re-encryption.

In addition, as modifications of the embodiment 16 and the embodiment 17, the tag ID information may be encrypted with a common key, and an encrypted text of the common key and tag ID information may be encrypted with the public key according to the above mentioned public key encryption technique to provide privileged ID information (hybrid encryption). In this instance, the security server acquires the common key by decrypting the privileged ID information with the secret key which corresponds to this public key, and decrypts the encrypted text of tag ID information using this common key to obtain tag ID information. Subsequently, the security server generates a separate encrypted text form this tag ID information according to the common key encryption technique, and the common key and the encrypted text may be further encrypted according to the public encryption technique. The resulting encrypted text may be used as a new privileged ID information to be returned to a client apparatus. Subsequently, this new privileged ID information is stored in the confidential value memory of the tag device in the similar as in the embodiment 16 or the like.

[0184] [Embodiment 18]

In an embodiment 18, a security server is changed at the time the privileged ID information is updated. A distinction over the embodiment 14 will be principally described below.

Fig. 38 is a conceptual view illustrating an overall arrangement of an updating system 2400 of the present embodiment. It is to be noted that in Fig. 38, arrangements which are common with the embodiment 14 are designated by like characters as used in the embodiment 14.

As illustrated in this Figure, the updating system 2400 comprises a tag device 2410, a client apparatus 2020 (equivalent to “update solicitor”), a

plurality of security servers 2460-1 ~ v (equivalent to “updaters”) and a backend apparatus 2050, which are connected together to enable a commutation therebetween through the network 2070.

[0185] Fig. 39 is a block diagram illustrating a functional arrangement of an updating system 2400 of the present embodiment, and Fig. 40 is a flow chart for describing a processing procedure therefor. A functional arrangement and a processing according to the present embodiment will be described below with reference to these Figures. It is to be noted that in Fig. 39, arrangements which are common with the embodiment 14 are designated by like characters as used in the embodiment 14. For brevity of illustration, only two security servers 2460-1 and 2460-2 are shown in Figs. 38 and 39, but it should be understood that more than two security servers may be used to construct a system. In addition, while only processing function/data which are required for purpose of description are indicated in Fig. 39, it should be understood that either one of the security servers 2460-1 and 2460-2 may also contain processing functions and data which the other security server contains. The security servers 2460-1 and 2460-2 execute respective processings under the control of controllers 2465-1 and 2465-2.

[0186] <pre-processing>

Privileged ID information according to this mode is information including an encrypted text according to the public key encryption technique and key ID information which corresponds to the public key used in the encryption. In the present example, privileged ID information of the tag device 2410 is defined as $sid_h = (epk_j(id_h), kid_j)$.

This privileged ID information $sid_h = (epk_j(id_h), kid_j)$ is stored in the confidential value memory 2411 of the tag device 2410 of this example. Each key ID information (kid_1, \dots, kid_n) and a secret key (sk_1, \dots, sk_n) of the

public key encryption technique are stored in a key memory 2461-1 of the security server 2460-1 in a manner relating to each other. Also, each key ID information (kid_1, \dots, kid_n) and a public key (pk_1, \dots, pk_n) of the public key encryption technique are stored in a key memory 2461-2 of the security server 2460-2 in a manner relating to each other.

[0187] In this example also, an identical key ID information is allotted to unrelated tag devices. This allows the variety of goods or a particular one of the goods to be prevented from being identified on the basis of the key ID information.

10 <privileged ID updating processing>

In the similar as in the embodiment 14, a client apparatus 2020 initially transmits a read command to tag device 2410 (step S380). The tag device 2410 extracts the privileged ID information ($sid_h=(epk_j(id_h), kid_j)$) from the confidential value memory 2411 (step S381) and transmits it to the client apparatus 2020 (step S382). In response thereto, the client apparatus 2020 causes its communication section 2021 (equivalent to “first ID output section”) to transmit (deliver) the privileged ID information sid_h extracted from the tag device 2410 together with a solicitation to update to the security server 2460-1 (step 383). It is to be noted that the security server 2460-1 is the security server which controls the privileged ID information stored in the tag device 2410 at this point in time.

[0188] Information inclusive of the privileged ID information sid_h is received by a communication section 2462-1 (equivalent to “first input section”) of the security server 2460-1 (step S384), and $epk_j(id_h)$ which constitutes the privileged ID information sid_h is sent to an ID extractor 2466-1 while kid_j is sent to a read/write section 2464-1. Upon receiving the key ID information kid_j , the read/write section 2464-1 extracts a secret key sk_j which

corresponds to the key ID information kid_j from the key memory 2461-1 and sends it to the ID extractor 2466-1 (step S385). Upon receiving the secret key sk_j , the ID extractor 2466-1 decrypts an encrypted text $epk_j(id_h)$ using the secret key sk_j to determine tag ID information id_h ($id_h = dsk_j(epk_j(id_h))$) (step S386). The tag ID information id_h which is thus determined is sent to a communication section 2462-1 (equivalent to “second output section”), and thence transmitted (delivered) to the client apparatus 2020 through the network 2070 (step S387).

[0189] The tag ID information id_h delivered from the security server 2460-1 is received by the communication section 2021 of the client apparatus 2020 (accepted as an input) (step S388). Subsequently, the communication section 2021 transmits (delivers) the tag ID information id_h to an arbitrarily selected security server 2460-2 to solicit to update the privileged ID information (step S389).

The communication section 2462-2 (equivalent to “third input section”) of the security server 2460-2 receives this tag ID information id_h (accepts as an input) which is transmitted through the network 2070 and sends it to an encryptor 2467-2 (step S390). This triggers a key selection by a key selector 2468-2, and such information is sent to a read/write section 2464-2 (step S391). In the present example, the key selector 2468-2 selects an arbitrary key number i (such as a random number) from natural numbers equal to or greater than 1 and equal to or less than n , and sends this key number i to the read/write section 2464-2. The read/write section 2464-2 extracts key ID information kid_i and the public key pk_i which correspond to the received key number i from the key memory 2461-2, and sends them to the encryptor 2467-2 (step S392). The encryptor 2467-2 encrypts (privileges) the tag ID information id_h using the received public key pk_i ($epk_i(id_h)$), and

generates a new privileged ID information ($sid_h' = (epk_i(id_h), kid_i)$) which comprises the encrypted text and the key ID information kid_i (step S393). The generated privileged ID information sid_h' is sent to the communication section 2462-2, and the communication section 2464-2 (equivalent to “third output section”) transmits (delivers) this privileged ID information sid_h' to the client apparatus 2020 through the network 2070 (step S394).

[0190] The client apparatus 2020 receives the privileged ID information (sid_h') (accepts as an input) by its communication section 2021 (equivalent to “second input section”) (step S395), and transmits it through the interface 2022 to the tag device 2410 (step S396). The tag device 2460 stores the new privileged ID information sid_h' in the confidential value memory 2411 (step S397), and responds with the new privileged ID information sid_h' in response to a subsequent read demand. Hereafter, the security server 2460-2 acts as the security server which controls the privileged ID information stored in the tag device 2410. Accordingly, a decryption of the new privileged ID information sid_h' takes place in the security server 2460-2 subsequently. Tag ID information id_h which represents a result of decrypting operation will be sent to the client apparatus 2020, the backend apparatus 2050 or the like. A decryption of the privileged ID information sid_h' in the security server 2460-2 takes place using the secret key sk_i stored in a key memory 2461-2 (a secret key corresponding to kid_i , not shown).

[0191] <features of embodiment 18>

It will be seen that in the embodiment 18, privileged ID information is decrypted by the security server 2460-1 which controls the privileged ID information in the tag device 2410, and the security server 2460-2 which is separate generates new privileged ID information to update the privileged ID information stored in the tag device 2410. In other words, an updating of the

privileged ID information takes place simultaneously with changing the security server which controls the privileged ID information in the tag device 2410. This prevents a concentration of information relating to a history of updating the privileged ID information in a single security server, allowing the risks of information leaking from the security server and of a tort by a security server which is set up with a malignant intention to be alleviated. By making the changed security server to be a local one which the public has no access, a higher level of safety can be realized.

[0192] The updating system of the present embodiment may be constructed with the common key encryption technique rather than the public key encryption technique.

Alternatively the updating system of the present embodiment may be constructed by applying the mode such as the embodiment 14 in which a random value is used in the privileged ID information. In this instance, the new security server generates a random value in place of the encryption mentioned above, and adds the generated random value (privileged ID) and ID anew to the privileged ID memory as used in the embodiment 14.

[Embodiment 19]

In an embodiment 19, re-privileging processing of the privileged ID information is performed by a client apparatus. Thus the client apparatus functions as an updater. In this instance, the client apparatus performs a re-privileging processing of the privileged ID information which is directly read out.

[0193] Fig. 41 is a block diagram illustrating a functional arrangement of an updating system 2500 of the present embodiment and Fig. 42 is a flow chart for describing a processing procedure therefor. A functional arrangement and a processing according to the present embodiment will be described below

with reference to these Figures. It should be noted that characters as used in the embodiment 14 are applied to arrangements in Fig. 41 which are common with the embodiment 14. A distinction over the embodiment 14 will be principally described below.

5 <pre-processing>

The privileged ID information of the present embodiment is an information including an encrypted text formed with an encryption algorithm (the public key encryption technique) having the property of re-encryption, and the key ID information which corresponds to the public key used in the encryption. In the present example, ElGamal encryption is used, and the privileged ID information in a tag device 2510 is defined as

$$\text{sid}_h = (g^r, \text{id}_h \cdot \text{pk}_j^r, \text{kid}_j).$$

[0194] This privileged ID information ($\text{sid}_h = (g^r, \text{id}_h \cdot \text{pk}_j^r, \text{kid}_j)$) is stored in the memory 2511 of the tag device 2510 of this example. Each key ID information ($\text{kid}_1, \dots, \text{kid}_n$) and a public key ($\text{pk}_1, \dots, \text{pk}_n$) are stored in the key memory 2524 of the client apparatus 2520 (equivalent to “updater”) in a manner relating to each other. An element of generation g is stored in a memory of a remainder multiplication calculator 2527.

Also in this example, an identical key ID information is allotted to unrelated tag devices. This prevents the variety of goods or a particular one of goods from being identified on the basis of the key ID information.

[0195] <privileged ID updating processing>

The client apparatus 2520 executes processings indicated below under the control of a controller 2023.

25 In the similar manner as in the embodiment 14, initially the client apparatus 2520 transmits a read command to the tag device 2510 (step S400). The tag device 2510 extracts the privileged ID information

$(sid_h = (g^r, id_h \cdot pk_j^r, kid_j))$ from the memory 2511 (step S401), and transmits it to the client apparatus 2520 (step S402).

This privileged ID information sid_h is received by the interface 2022 of the client apparatus 2520 (step S403), and an encrypted text $(g^r, id_h \cdot pk_j^r)$

5 which constitute the privileged ID information sid_h is sent to the remainder multiplication calculator 2528 (which constitutes an “encryptor”) while kid_j is sent to a read/write section 2525. kid_j is also recorded in a memory 2023a.

[0196] Upon receiving the key ID information kid_j , the read/write section 2525 extracts the public key pk_j which corresponds to this key ID information

10 kid_j from the key memory 2524, and sends it to a remainder power calculator 2527 (which constitutes an “encryptor”) (step S404). This triggers the random number generator 2526 to generate a random number r' which is equal to or greater than 0 or equal to or less than $p-1$, which is then sent to the remainder power calculator 2527 (step S405). The remainder power calculator 2527

15 calculates $(g^{r'}, pk_j^{r'})$ using the element of generation g within its own memory, the received public key (pk_j) and the random number r' , and sends its result to the remainder multiplication calculator 2528 (step S406). The remainder multiplication calculator 2528 calculates $(g^{r+r'}, id_h \cdot pk_j^{r+r'})$ using received $(g^{r'}, pk_j^{r'})$ and $(g^r, id_h \cdot pk_j^r)$, and sends a result of calculation as a new encrypted text

20 to the interface 2022 (step S407). The interface 2022 transmits (delivers) the encrypted text $(g^{r+r'}, id_h \cdot pk_j^{r+r'})$ which is sent thereto and key ID information kid_j in the memory of the interface 2022 as new privileged ID information $(sid'_h = (g^{r+r'}, id_h \cdot pk_j^{r+r'}, kid_j))$ (step S408).

[0197] The transmitted new privileged ID information sid'_h is received by
25 the interface 2013 of the tag device 2510, and is stored in the memory 2511 through the read/write section 2012 (step S409). Subsequently, in response to a read demand, the tag device 2510 responds with this new privileged ID

information sid_h' .

<features of embodiment 19>

In the embodiment 19, the client apparatus 2520 re-privileges the privileged ID information within the tag device 2510. The client apparatus
5 2520 applies the re-privileging processing only to the privileged ID information which is directly read out by the interface 2022. This suppresses a leakage of information to a third party, securing a higher level of safety.

[0198] While the public key (pk_1, \dots, pk_n) is stored in the key memory 2524 of the security server 2520 in the present embodiment, an arrangement
10 may be used in which the security server 2520 does not carry the public key (pk_1, \dots, pk_n) , but acquires the public key (pk_1, \dots, pk_n) from a given public key server for use.

The arrangement of the security server shown in either mode from the embodiment 14 to the embodiment 16 may be applied to the client
15 apparatus 2520 to execute processings of the present embodiment.

[Embodiment 20]

An embodiment 20 will now be described.

[0199] In this embodiment, a client apparatus (equivalent to “update solicitor”) previously acquires a plurality of privileged ID information, and
20 selects one therefrom to be used in updating the privileged ID information within the tag device.

Fig. 43 is a block diagram illustrating a functional arrangement of an updating system 2600 of the present embodiment, and Fig. 44 is a flow chart for describing a processing procedure therefor. A functional arrangement and
25 a processing of the present embodiment will be described below with reference to these Figures. It is to be noted that in Fig. 43, arrangements which are common with the embodiment 14 are designated by like characters

as used in the embodiment 14. A distinction over the embodiment 14 will be principally described below.

[0200] <pre-processing>

Initially, the communication section 2021 (equivalent to “privileged
5 ID input section”) of the client apparatus 2620 receives (accepts as input) a plurality of kinds of privileged ID information (sid_h-1, \dots, p) which are sent through the network 2070 (step S410). The plurality of kinds of privileged ID information (sid_h-1, \dots, p) are those which are obtained by repeating the method described in either one of the embodiment 14 to the embodiment 17
10 or by transmitting a plurality of kinds of privileged ID information in one operation from the security server 2660. When utilizing the method of the embodiment 14, it is necessary that a plurality of privileged ID information (sid_h-1, \dots, p) be stored in the privileged ID memory of the security server 2660 for one tag ID information. By contrast, when utilizing the method in
15 either one of the embodiment 15 to embodiment 17, information which is stored in the security server 2660 may be similar to one of the embodiment 15 to the embodiment 17.

[0201] The communication section 2021 sends these privileged ID
information (sid_h-1, \dots, p) to the read/write section 2624, which then store
20 them in the privileged ID memory 2625 (step S411).

<privileged ID updating processing>

The client apparatus 2620 executes the processings described below under the control of the controller 2023.

Initially, the controller 23 determines whether or not there existed a
25 given trigger (opportunity) to update the privileged ID information (step S412). What can be cited as such a trigger are that the privileged ID information has been read from the tag device 2610, that a count indicating a

number of times the privileged ID information within the tag device 2610 has been used has reached a given value or the like. In the absence of a given trigger, a determination rendered at step S412 is continued, and in the presence of a given trigger, the read/write section 2624 (equivalent to

5 “privileged ID extractor”) extracts one privileged ID information sid_{h-j} from the privileged ID memory 2625 (step S413). The selection of this one privileged ID information sid_{h-j} may take place at random, or may be in the sequence of an array in the manner of sid_{h-1} , sid_{h-2} , \dots and returning to sid_{h-1} again after sid_{h-p} . The extracted one privileged ID information sid_{h-j} is

10 sent from the read/write section 2624 to the interface 2022 (equivalent to “privileged ID output section”), and thence transmitted (delivered) to the tag device 2610 (step S414).

[0202] The tag device 2610 receives this privileged ID information sid_{h-j} at its interface 2013 (step S415), and stores it in a confidential value memory

15 2611 through the read/write section 2012 (step S416). Subsequently, the tag device 2610 responds with this new privileged ID information $sid_{h'}$ in response to a read command from the reader.

<features of embodiment 20>

In the present embodiment, a plurality of kinds of privileged ID

20 information are previously stored in the client apparatus 2620, and the privileged ID information in the tag device 2610 is updated by the privileged ID information which is selected from the stored ones. The selection of the privileged ID information which is used in updating takes place within the client apparatus 2620, and its transmission takes place locally between the

25 client apparatus 2620 and the tag device 2610. Accordingly, a leakage of information to a third party can be suppressed, securing a higher level of safety. If the transmission of the plurality of kinds of privileged ID

information takes place in one operation from the security server 2660 to the client apparatus 2620, a number of times the security server 2660 is accessed can be reduced, thus alleviating a degradation in the performance of the system which is associated with the updating processing of the privileged ID
5 information.

[0203] It should be noted that the opportunity for selecting/storing the privileged ID information is not limited to the one mentioned above, and an arrangement may be used in which after the privileged ID information stored in the client apparatus 2620 have been exhausted, a plurality of kinds of
10 privileged ID information may be acquired again from the security server 2660 to be stored in the client apparatus 2620.

[Embodiment 21]

An embodiment 21 will now be described.

This embodiment is a modification of the embodiment 20, and differs
15 from the embodiment 20 in that the client apparatus acquires the privileged ID information delivered from a plurality of security servers (“updaters”).

[0204] Fig. 45 is a block diagram illustrating a functional arrangement of an updating system 2770 of the present embodiment. A functional arrangement and a processing of the present embodiment will be described below with
20 reference to this Figure. It is to be noted that in Fig. 45, arrangements which are common with the embodiment 14 or the embodiment 20 are designated by like character as used in the embodiment 14 or the embodiment 20. Only a distinction over the embodiment 20 will be described below.

<pre-processing>

25 The only difference over the embodiment 20 resides in that a client apparatus 2620 receives a plurality of kinds of privileged ID information (sid_h-1, \dots, p) which are sent from a plurality of security servers 2760-1,

2760-2, ..., 2760-p. Privileging of ID in the plurality of security servers 2760-1, 2760-2, ..., 2760-p takes place according to the method of the embodiment 18, for example.

[0205] <privileged ID updating processing>

5 This remains to be similar to the embodiment 20.

<features of embodiment 21>

In the present embodiment, the client apparatus 2620 acquires the privileged ID information which is generated by the plurality of security servers 2760-1, 2760-2, ..., 2760-p. Accordingly, a concentration of the
10 history of updating the privileged ID information in one security server can be prevented, realizing a higher level of safety.

[0206] As mentioned above, when the method of the embodiment 14 is utilized to generate privileged ID information in the embodiment 20, there has been a need to hold a plurality of privileged ID information (sid_h-1, \dots, p) for
15 one key ID information in the privileged ID memory of the security server. However, in the present embodiment, even though the method of the embodiment 14 is utilized to generate privileged ID information, the privileged ID information which is controlled by each security server may be only one privileged ID information for one key ID information. In this respect,
20 a control of the privileged ID information can be simplified.

[Embodiment 22]

An embodiment 22 will now be described.

[0207] This embodiment is a modification of the embodiment 20 and the embodiment 21, and in this modification, a plurality of privileged ID
25 information which are acquired is stored in a tag device rather than in a client apparatus.

Fig. 46 is a block diagram illustrating a functional arrangement of an

updating system 2800 of the present embodiment. A functional arrangement and a processing of the present embodiment will be described below with reference to this Figure. It is to be noted that in Fig. 46, arrangements which are common with the embodiment 14 are designated by like characters as used in the embodiment 14. A distinction over the embodiment 14, the embodiment 20 and the embodiment 21 will be principally described below.

<pre-processing>

Initially, a communication section 2021 of a client apparatus 2020 receives a plurality of kinds of privileged ID information (sid_h-1, \dots, p) which are sent through the network 2070. The plurality of kinds of privileged ID information (sid_h-1, \dots, p) which are received are sent to an interface 2022, and thence transmitted to a tag device 2810.

[0208] The tag device 2810 receives (accepts as input) the plurality of kinds of privileged ID information (sid_h-1, \dots, p) at its interface 2013 (equivalent to “privileged ID input section”), and sends them to a read/write section 2012, which then stores them in a privileged ID memory 2811. It is to be noted that the privileged ID information (sid_h-1, \dots, p) may be ones which are sent from a single security server or ones which are delivered from a plurality of security servers.

<privileged ID updating processing>

The read/write section 2012 of the tag device 2810 (equivalent to “privileged ID extractor”) extracts one privileged ID information (sid_h-j) arbitrarily (for example, at random) from the privileged ID memory 2811 under the control of a controller 2014 in response to a trigger (opportunity) which may be a read command from a reader, for example, and transmits it from an interface 2013. The transmitted privileged ID information (sid_h-j) is used in a processing by the backend apparatus as described above in

connection with the embodiment 14.

[0209] <features of embodiment 22>

In the present embodiment, a plurality of kinds of privileged ID information (sid_h-1, \dots, p) are stored in the tag device 2810, and one
5 privileged ID information (sid_h-j) which is selected therefrom is used. This avoids that the privileged ID information which is used in the acquisition of information relating to ID be identical with each other during each run, thus allowing a tracing of the tag device 2810 to be suppressed. Since the plurality of kinds of privileged ID information (sid_h-1, \dots, p) are stored in the tag
10 device 2810 itself, the privileged ID information which is used can be updated if an access to the client apparatus 2020 cannot be made (as when a reading processing is performed in a tag reader which does not have the function of a client apparatus 2020, for example).

[0210] [Embodiment 23]

15 In this embodiment, a tag device is provided with a confidential value memory having a read-only region in which key ID information is stored and a rewritable region in which a first privileged ID information is stored. When re-privileging processing of the privileged ID information is made, key ID information and the first privileged ID information are extracted
20 from the privileged ID memory to be delivered.

An updater accepts the key ID information and the first privileged ID information as inputs, and extracts a key which corresponds to the key ID information. Using the extracted key and the first privileged ID information, it generates a second privileged ID information, the association with the first
25 privileged ID information of which is difficult to follow, and this second privileged ID information is delivered.

[0211] A tag device accepts the second privileged ID information as an

input and stores the second privileged ID information which is input in the rewriteable region of the privileged ID memory.

What is updated by the updater is only the privileged ID information. And what is rewritten in the tag device is only the privileged ID information in the rewritable region, and there is no change in the key ID information in the read-only region. Accordingly, if the privileged ID information in the rewritable region is rewritten to a privileged ID information which corresponds to a different tag device, the key ID information which is used in a decrypting processing of the privileged ID information remains unchanged from the original key ID information. For this reason, a decrypting server which is selected when decrypting the rewritten privileged ID information is a decrypting server which is selected on the basis of the original key ID information, for example, and it is possible that a decrypting processing of the rewritten privileged ID information may not be appropriately achieved. If the decrypting server were in common, the key which is used in the decrypting processing of the rewritten privileged ID information is the key which corresponds to the original key ID information, and accordingly, a decrypted result may become extraordinary.

[0212] This embodiment will be described below with reference to the drawings.

Fig. 47 is a conceptual view illustrating an overall arrangement of an updating system 3000 of this embodiment.

As illustrated in this Figure, an updating system 3000 comprises a tag device 3010 such as a radio tag or the like which is applied to goods, a client apparatus 3020, a backend apparatus 3050 which controls products distribution information or the like which relates to ID in plain text, a security server 3060 which performs a re-privileging processing of privileged ID

information, and a security server 3070 which performs a restoration processing of ID. The client apparatus 3020, the backend apparatus 3050 and the security servers 3060 and 3070 are connected together to enable a communication therebetween through a network 3080 such as internet or the like. For brevity of description, in this Figure, the tag device 3010, the client apparatus 3020, the backend apparatus 3050 and the security servers 3060 and 3070 are each shown for only one, but it is to be noted that a plurality of tag devices are normally provided and that more than one client apparatus, backend apparatus and security servers may also be provided. In addition, the security servers 3060 and 3070 may be replaced by a security server which has the both functions of the security servers 3060 and 3070.

[0213] In the client apparatus 3020 of this example, it reads privileged ID information from the tag device 10 and sends it to the security server 3070. The security server 3070 restores ID from the privileged ID information, and returns this ID to the client apparatus 3020. Upon receiving ID, the client apparatus 3020 accesses the backend apparatus 3050 to demand an entry of information including ID, a date and time of reading, a location of reading, a temperature or the like and an acquisition of information relating to ID. Also, a mode of utilizing a proxy model may be contemplated in which the client apparatus 3020 transmits the privileged ID information to the security server 3070, which then directly accesses the backend apparatus 3050.

[0214] At a given opportunity, the privileged ID information within the tag device 3010 is subjected to re-privileging processing in the security server 3060 (namely, updating the privileged ID information into a separate privileged ID information), whereby the privileged ID information within the tag device 3010 is updated. In order to secure an opportunity to update the privileged ID information in a positive manner, the client apparatus 3020 may

be located at the porch of a house. In this instance, each time a user who carries the tag device passes through the porch, the client apparatus 3020 reads the privileged ID information within tag device 3010, which is re-privileged by the security server 3060 to be written into the tag device again.

[0215] The present embodiment is characterized in that the tag device 3010 is provided with privileged ID memory including a read-only region in which key ID information is stored and a rewritable region in which privileged ID information is stored, and that while the re-privileged privileged ID information is written into the rewritable region, the read-only region in which the key ID information is stored is not updated. It is to be noted that the privileged ID information which is stored in the rewritable region does not contain key ID information.

<functional arrangement/processing>

Fig. 48 is an illustration of a functional arrangement of an updating system 3000 of the present embodiment, and Figs. 49 and 50 are flow charts for describing a processing procedure therefore. A functional arrangement and a processing of the present embodiment will be described below with reference to these Figures. A description of the backend apparatus will be omitted in connection Fig. 48 and subsequent Figures. The tag device 3010, the client apparatus 3020 and the security servers 3060 and 3070 execute respective processings under the control of controllers 3014, 3023, 3065 and 3075, respectively. Data which are processed are sequentially stored in memories 3014a, 3023a, 3065a or 3075a, and is accessed when a processing such as a calculation is performed, but will be omitted from the description to follow.

[0216] <pre-processing>

In this mode, an encrypted text formed according to an encryption algorithm having the property of re-encryption (public key encryption technique) is used as privileged ID information. In this example, elliptical ElGamal encryption is used.

5 As shown in Fig. 48, the tag device 3010 of this example has a confidential value memory 3011 including a read-only region 3011a and a rewritable region 3011b. As a confidential value memory 3011, a rewritable memory such as rewritable ROM (Read Only Memory) such as EEPROM may be used, and given regions thereof may be allotted to the read-only
10 memory 3011a and the rewritable region 3011b. Alternatively, a memory such as ROM which cannot be rewritten may be used to construct the read-only region 3011a while a rewritable memory such as EEPROM may be used to construct the rewritable region 3011b. Key ID information kid_j which specifies a secret key sk_j and a public key pk_j is stored (recorded) in the
15 read-only region 3011a, and privileged ID information $sid_h=(g^r, id_h \cdot pk_j^r)$ is stored in the rewritable region 3011b.

[0217] An element of generation g is stored in the memory 3065a of the security server 3060 (“updater”), and each key ID information (kid_1, \dots, kid_n), a secret key (sk_1, \dots, sk_n) and a public key (pk_1, \dots, pk_n)
20 are stored in the key memory 3071 of the security server 3070 (equivalent to “decryptor”) in a manner relating to each other.

In this example, it is assumed that a total number m of tag devices is sufficiently greater than a total number n of keys ($m \gg n$), and an identical key ID information is allotted to unrelated tag devices. Specifically, rather than
25 allotting an identical key ID information to tag devices which are applied to goods of the same variety, for example, an identical key ID information is allotted to tag devices which are applied to unrelated goods. This allows

preventing tag devices, the variety of goods or a particular one of goods from being uniquely identified on the basis of the key ID information.

[0218] <privileged ID decrypting processing>

Initially, a decrypting processing of privileged ID information which
5 is performed when demanding a backend apparatus 50 to acquire information relating to ID will be described.

Initially, utilizing some sort of authentication technology, a mutual authentication is performed between the client apparatus 3020 and the security server 3070. It is to be noted that a communication between the client
10 apparatus 3020 and the security server 3070 takes place by using an encryption according to some encryption technique.

The client apparatus 3020 then transmits a read command to the tag device 3010 from its interface 3022 (step S501). This read command is received by an interface 3013 of the tag device 3010, and this triggers a
15 read/write section 3012 to extract key ID information kid_j from the read-only region 3011a of the confidential value memory 3011 and to extract privileged ID information sid_h from the rewritable region 3011b (step S502). The extracted privileged ID information sid_h and the key ID information kid_j are transmitted to the client apparatus 3020 through the interface 3013 (step
20 S503), and are received by the interface 3022 of the client apparatus 3020. The client apparatus 3020 identifies an address of the security server 3070 from the received key ID information kid_j , for example, and transmits the privileged ID information sid_h and key ID information kid_j to the security server 3070 from its communication section 3021 through a network 3080
25 (step 504).

[0219] The transmitted privileged ID information sid_h and key ID information kid_j are received by a communication section 3072 (equivalent to

“privileged ID input section”) of the security server 3070 (accepted as inputs) (step S505), and the privileged ID information sid_h is fed to a decryptor 74 (equivalent to “ID calculator”) while the key ID information kid_j is fed to a read/write section 3073. The read/write section 3073 (equivalent to “key
5 extractor”) extracts a secret key sk_j which corresponds to the key ID information kid_j which is sent thereto from a key memory 3071, and sends it to the decryptor 3074 (step S506). The decryptor 3074 calculates a tag ID information id_h which is decrypted from the privileged ID information sid_h using the privileged ID information sid_h and the secret key sk_j which are sent
10 thereto. In this example, the tag ID information id_h is calculated by a calculation of $id_h = (id_h \cdot pk_j^r) / (g^r)^{sk_j}$. It is to be noted that the index “ sk_j ” in this calculation formula means “ sk_j ”. The calculated tag ID information id_h is sent to the communication section 3072, which then transmits it toward the client apparatus 3020 through the network 3080 (step S508). The client apparatus
15 3020 receives the transmitted tag ID information id_h at its communication section 3021 (step S509), and utilizes this tag ID information id_h for a subsequent inquiry to the backend apparatus 3050.

[0220] <privileged ID updating processing>

Privileged ID information updating processing in the present
20 embodiment will now be described.

Initially, using some sort of authentication technology, a mutual authentication is made between the client apparatus 3020 and the security server 3060. It is to be noted that a communication between the client apparatus 3020 and the security server 3060 takes place in an encrypted form
25 according to some encryption technique.

Privileged ID information updating processing in this example is initiated at an arbitrary opportunity such as a passage of a user through a

location such as a front porch which he never fails to pass when leaving the house or at a number of uses of privileged ID information stored in the tag device 3010 (such as a count reaching a given value), which acts as a trigger. In response to this trigger, the client apparatus 3020 transmits a read

5 command to the tag device 3010 from its interface 3022 (step S511). This read command is received by the interface 3013 of the tag device 3010, and this triggers the read/write section 3012 (equivalent to “privileged ID extractor”) to extract key ID information kid_j from the read-only region 3011a of the confidential value memory 3011 and to extract privileged ID
10 information sid_h from the rewritable region 3011b (step S512). The extracted privileged ID information sid_h and key ID information kid_j are transmitted (delivered) through the interface 3013 (equivalent to “privileged ID extractor”) to the client apparatus 3020 (step S513), and are received by the interface 3022 of the client apparatus 3020, which then transmits the received
15 privileged ID information sid_h and key ID information kid_j to the security server 3060 through the communication section 3021 and the network 3080 (step 514).

[0221] The security server 3060 receives (accepts as inputs) the privileged ID information sid_h and the key ID information kid_j at its communication
20 section 3061 (equivalent to “privileged ID input section”) (step S515), and sends the privileged ID information ($sid_h = (g^f, id_h \cdot pk_j^f)$) to the remainder multiplication calculator 3064 (which constitutes “privileged ID updating section”). The communication section 3061 (equivalent to “key extractor”) transmits this key ID information kid_j together with a demand to acquire a
25 public key to the security server 3070 through the network 3080.

The security server 3070 receives them at its communication section 3072, and sends the key ID information kid_j to the reader 3073. The reader

3073 extracts a public key pk_j which corresponds to this key ID information kid_j from the key memory 3071, and returns the extracted public key pk_j to the security server 3060 through the communication section 3072 and the network 3080.

- 5 [0222] The security server 3060 receives (extracts) this public key pk_j at its communication section 3061, and sends it to the remainder power calculator 3063 (which constitutes “privileged ID updating section”) (step S516). This triggers, for example, a random number generator 3062 to generate a random number r' which is equal to or greater than 0 and equal to or less than $p-1$ and
- 10 to send it to the remainder power calculator 3063 (step S517). The remainder power calculator 3063 calculates $(g^{r'}, pk_j^{r'})$ using the element of generation g within the memory 3065a, the received public key pk_j and the random number r' , and sends a result of the calculation to the remainder multiplication calculator 3064 (step S518). The remainder multiplication calculator 3064
- 15 calculates $(g^{r+r'}, id_h \cdot pk_j^{r+r'})$ using received $(g^{r'}, pk_j^{r'})$ and $(g^r, id_h \cdot pk_j^r)$, and sends a result of this calculation (an encrypted text) as new privileged ID information to the communication section 3061 (step S519). The communication section 3061 (equivalent to “privileged ID output section”) transmits (delivers) the privileged ID information $(sid_h' = (g^{r+r'}, id_h \cdot pk_j^{r+r'}))$
- 20 (privileged ID information sid' , the association of which with the privileged ID information (sid_h) before updating is difficult to follow) to the client apparatus 3020 through the network 3080 (step S520).

- [0223] The transmitted new privileged ID information sid_h' is received by the communication section 3021 of the client apparatus 3020, and is
- 25 transmitted from its interface 3022 to the tag device 3010 (step S521). The tag device 3010 receives (accepts as input) this new privileged ID information sid_h' at its interface 3013 (equivalent to “privileged ID input section”) (step

S522), and its read/write section 3012 (equivalent to “privileged ID storage”) stores this new privileged ID information sid_h' in the rewritable region 3011b of the memory 3011 (step S523). The tag device 3010 responds with this new privileged ID information sid_h' in response to a subsequent read demand.

5 [0224] <features of embodiment 23>

In the present embodiment, there is provided the confidential value memory 3011 having the read-only region 3011a in which key ID information is stored and the rewritable region 3011b in which the privileged ID information is stored, and only the privileged ID information stored in the
10 rewritable region 3011b is re-privileged to be updated. Accordingly, if privileged ID information of a separate tag device were written into the rewritable region 3011b when re-privileging the privileged ID information, it is possible to detect such an irregularity/mistake.

[0225] By way of example, referring to Fig. 48, an instance is considered in
15 which privileged ID information $(g^r, ID_2 \cdot pk_1^r)$ of a separate tag device which corresponds to key ID information kid_1 is stored in the rewritable region 3011b of the confidential value memory 3011 of the tag device 3010. Even in this instance, key ID information which is stored in the read-only region 3011a remains to be kid_j , and the secret key which is extracted from the key
20 memory 3071 by the reader 3073 during the decrypting processing of the security server 3070 is sk_j which corresponds to kid_j . Accordingly, a result of decryption from the decryptor 3074 is $(id_2 \cdot pk_1^r)/(g^r)^{sk_j} = (id_2 \cdot (g^{sk_1})^r)/(g^r)^{sk_j} = ID_2 \cdot g^{sk_1}/g^{sk_j}$, whereby a result of calculation indicates an extraordinary data. Accordingly, a decrypted result represented by extraordinary data allows a
25 detection of the fact that the privileged ID information of a separate tag device has been written into.

[0226] Since a wrong rewriting of privileged ID information can be

prevented without a control as by password of an access to the rewritable region 3011b, the cost for the control circuit can be suppressed while dispensing with a complex password control for purpose of an access control.

Specifically, this enables an updating of privileged ID information at an arbitrary timing to be executed in a more reliable and safe manner and at a reduced cost, permitting a protection of the privacy of the tag device 3010.

It is added that while privileged ID information is generated and updated using elliptical ElGamal encryption in the present embodiment, an encryption having the property of re-encryption or re-privileging technique disclosed in Patent Application No. 2003-359157 may also be used. In addition, the security servers 3060 and 3070 may be unified, and in addition, the security server 3060 may be provided with a memory for the public key.

[0227] <Embodiment 24>

The present embodiment is a modification of the embodiment 23, in which a confirmation is made to see whether or not a result of decrypting privileged ID information has any contradiction with the format of ID, thereby confirming whether or not privileged ID information which is delivered from the tag device is wrong. A distinction over the embodiment 23 will be principally described below while omitting a description of matters which are common with the embodiment 23.

Fig. 51 is an illustration of a functional arrangement of a security server 3170 (equivalent to “decryptor”) of the present embodiment, and Fig. 52 illustrates a format of tag ID information 3200 used in the present embodiment. Fig. 53 is a flow chart for describing a processing procedure by the security server 3170. It is to be noted that functional arrangements which are common with the embodiment 23 are designated in Fig. 51 by like characters as used in the embodiment 23.

[0228] <an overall arrangement/ hardware arrangement>

The arrangement is similar to the embodiment 23 except that the security server 3070 is replaced by the security server 3170.

<pre-processing>

5 A distinction over the embodiment 23 lies in that an effective value of each field of ID is stored in an effective value memory 3176 of the security server 3170. In other respects, the process is similar as in the embodiment 23.

[0229] <privileged ID decrypting processing>

A distinction over the embodiment 23 resides in substituting a
10 processing illustrated in Fig. 53 by the security server 3170 for the processing by the security server 3070 mentioned above (Fig. 49: steps S505 ~ S508). Only the processing by the security server 3170 will be described below while omitting a description for other processing.

In the similar manner as in the embodiment 23, privileged ID
15 information sid_h and key ID information kid_j transmitted from a client apparatus 3020 are received by a communication section 3072 (equivalent to “privileged ID input section”) of the security server 3170 (accepted as inputs) (step S531) and the privileged ID information sid_h is sent to a decrypting section 3074 (equivalent to “ID calculator”) while key ID information kid_j is
20 sent to a reader section 3073. The reader section 3073 (equivalent to “key extractor”) extracts a secret key sk_j which corresponds to the key ID information kid_j which is sent thereto, and sends it to the decrypting section 3074 (step S532). The decrypting section 3074 calculates the tag ID information id_h which is decrypted from privileged ID information sid_h using
25 privileged ID information sid_h which is sent thereto and the secret key sk_j .

[0230] The calculated tag ID information id_h is sent to an ID structure verifier 3177 where the structure of the tag ID information id_h is verified (step

S534). As illustrated in Fig. 52, ID3200 of this example has fields including a header (h) 3210, a version code (vc) 3202, a manufacturer code (mc) 3203, a products code (pc) 3204 and a serial code (sc) 3205. An effective value which each field can assume is stored in an effective value memory 3176, and the ID structure verifier 3177 compares a value of each field in the received tag ID information id_h against an effective value which is extracted from the effective value memory 3176 to verify whether or not the value of each field in the received tag ID information id_h is within a range for the effective value. Upon a successful verification (step S535), the ID structure verifier 3177 sends the tag ID information id_h to the communication section 3072, which then transmits the tag ID information id_h to the client apparatus 3020 (step S536). On the other hand, when the verification is unsuccessful (step S535), the ID structure verifier 3177 destroys the tag ID information id_h , terminating the processing operation.

15 [0231] <features of embodiment 24>

In the present embodiment, an arrangement is made for the ID structure verifier 3177 of the security server 3170 to verify whether or not the decrypted tag ID information id_h has no contradiction with a given ID format. This allows extraordinary which may result from decrypting a privileged ID information which is attributable to a writing of privileged ID information of a different tag device into the rewritable region of the tag device to be discovered in a reliable manner.

[Embodiment 25]

The present embodiment is a modification of the embodiment 23, and a distinction over the embodiment 23 resides in that during a re-privileging processing of privileged ID information, a secret key is applied to the key ID information and re-privileged privileged ID information to

attach authentication information such as a digital signature, MAC or the like. A distinction over the embodiment 23 will be principally described below while omitting a description for other matters which are common with the embodiment 23.

5 [0232] Figs. 54 and 55 illustrate functional arrangements of an updating system 3300 of the present embodiment, and Figs. 56 and 57 are flow charts for describing a processing procedure. It is to be noted that in Figs. 54 and 55, functional arrangements which are common with the embodiment 23 are designated by like characters as used in the embodiment 23.

10 <an overall arrangement/ hardware arrangement>

The arrangement is similar to the embodiment 23 except that a tag device 3310 is substituted for the tag device 3010, a security server 3360 (equivalent to “updater”) is substituted for the security server 3060, and a security server 3370 (equivalent to “decryptor”) is substituted for the security
15 server 3070.

[0233] <pre-processing>

Distinctions over the embodiment 23 reside in that privileged ID information sid_h and a digital signature (equivalent to “verification information”) are stored in a rewritable region 3311b of a confidential value
20 memory 3311 of the tag device 3310 and that the secret key sk_j and the public key pk_i used in the digital signature are stored in a key memory 3366 of the security server 3360. In other respects, the arrangement is similar to the embodiment 23.

<privileged ID updating processing>

25 Privileged ID information updating processing of the present embodiment will now be described.

[0234] Initially, a client apparatus 3020 transmits a read command to the tag

device 3310 from its interface 3022 (step S541). The read command is received by an interface 3013 of the tag device 3310, and this triggers a read/write section 3012 to extract key ID information kid_j from a read-only region 3011a of the confidential value memory 3311 and to extract privileged ID information sid_h from the rewritable region 3311b (step S542). The extracted privileged ID information sid_h and key ID information kid_j are transmitted to the client apparatus 3020 through the interface 3013 (step S543), and are received by the interface 3022 of the client apparatus 3020. The client apparatus 3020 transmits the received privileged ID information sid_h and key ID information kid_j to the security server 3360 through a communication section 3021 and a network 3080 (step 544).

[0235] The security server 3360 receives the privileged ID information sid_h and the key ID information kid_j at its communication section 3061 (step S545), and sends the privileged ID information ($sid_h=(g^r, id_h \cdot pk_j^r)$) to a remainder multiplication calculator 3064. In the similar as in the embodiment 23, the communication section 3061 transmits the key ID information kid_j to the security server 3370 where the extracted public key pk_j is acquired (received) (step S546). This public key pk_j is sent to a remainder power calculator 3063, and a random number r' which is generated in a random number generator 3062 (step S547) is also sent to the remainder power calculator 3063. The remainder power calculator 3063 calculates $(g^{r'}, pk_j^{r'})$, and sends a result of calculation to the remainder multiplication calculator 3064 (step S548), which then calculates $(g^{r+r'}, id_h \cdot pk_j^{r+r'})$, and sends this result of calculation as new privileged ID information to the communication section 3061 and a signature generator 3368 (step S549). This triggers a reader 3367 to extract a secret key sk from the key memory 3366 and to send it to the signature generator 3368 (step S550). The signature generator 3368

(equivalent to “verification information generator”) also receives the key ID information kid_j from the communication section 3061 to generate a bit combination data ($g^{r+r'} \mid id_h \cdot pk_j^{r+r'} \mid kid_j$) of $g^{r+r'}$, $id_h \cdot pk_j^{r+r'}$ and kid_j , for example, and generates a digital signature (equivalent to “verification information”) $\sigma' = E_{sk}(g^{r+r'} \mid id_h \cdot pk_j^{r+r'} \mid kid_j)$ which is formed by encrypting the bit combination data with the secret key sk (step S551). The generated new digital signature σ' is sent to the communication section 3061 (equivalent to “privileged ID output section”), which transmits (delivers) the new privileged ID information ($sid_h' = (g^{r+r'}, id_h \cdot pk_j^{r+r'})$) which is previously sent thereto and the new digital signature σ' to the client apparatus 3020 through the network 3080 (step S552).

[0236] The new privileged ID information sid_h' and the digital signature σ' which have been transmitted are received by the communication section 3021 of the client apparatus 3020 and are transmitted from the interface 3022 to the tag device 3310 (step S553). The tag device 3310 receives (accepts as inputs) the new privileged ID information sid_h' and the digital signature σ' at its interface 3013 (equivalent to “privileged ID input section”) (step S554), and the read/write section 3012 (equivalent to “privileged ID storing section”) stores the new privileged ID information sid_h' and the digital signature σ' in the rewritable region 3311b of the confidential value memory 3311 (step S555). Subsequently, the tag device 3310 responds with the new privileged ID information sid_h' and the digital signature σ' in response to a read demand.

[0237] <privileged ID decrypting processing>

Privileged ID information decrypting processing according to the present embodiment will now be described.

Initially, the client apparatus 3020 transmits a read command to the tag device 3310 from its interface 3022 (step S561). This read command is

received by the interface 3013 of the tag device 3310, and this triggers the read/write section 3012 to extract key ID information kid_j from the read-only region 3011a of the confidential value memory 3311 and to extract privileged ID information sid_h' and digital signature σ' from the rewritable region 3311b (step S562). The extracted privileged ID information sid_h' , digital signature σ' and key ID information kid_j are transmitted to the client apparatus 3020 from the interface 3013 (step S563), and are received by the interface 3022 of the client apparatus 3020. The client apparatus 3020 transmits such information from the communication section 3021 to the security server 3370 through the network 3080 (step 564).

[0238] The transmitted privileged ID information sid_h' , digital signature σ' and key ID information kid_j are received by the communication section 3072 (equivalent to “privileged ID input section”) of the security server 3370 (accepted as inputs) (step S565), the digital signature σ' is fed to a signature verifier 3376, the privileged ID information sid_h is fed to a decryptor 3074 (equivalent to “ID calculator”) and the signature verifier 3376, and the key ID information kid_j is fed to a reader 3073 and the signature verifier 3376.

The communication section 3072 also sends a public key acquisition demand to the security server 3360 through the network 3380, and the security server 3360 receives it at its communication section 3061 and causes its reader 3367 to extract the public key pk from the key memory 3363 and returns the public key pk through the communication section 3061 and the network 3080. The public key pk is received by the communication section 3072 of the security server 3370 (step S566) and is then sent to the signature verifier 3376.

[0239] The signature verifier 3376 decrypts the received digital signature σ' using the public key pk ($D_{pk}(\sigma')$), and generates a bit combination data ($g^{r+r'}$ |

$id_h \cdot pk_j^{r+r'} \mid kid_j$) of $g^{r+r'}$, $id_h \cdot pk_j^{r+r'}$ and kid_j . It then verifies the digital signature σ' by seeing whether or not $D_{pk}(\sigma')$ is equal to $(g^{r+r'} \mid id_h \cdot pk_j^{r+r'} \mid kid_j)$ (step S567). In the event $D_{pk}(\sigma') = (g^{r+r'} \mid id_h \cdot pk_j^{r+r'} \mid kid_j)$ does not apply, the processing is terminated as a failure of verification. On the other hand, if

- 5 $D_{pk}(\sigma') = (g^{r+r'} \mid id_h \cdot pk_j^{r+r'} \mid kid_j)$ does apply, a read/write section 3073 (equivalent to “key extractor”) extracts a secret key sk_j which corresponds to the key ID information kid_j which is sent thereto from a key memory 3071, and sends it to the decryptor 3074 (step S568). Using the privileged ID information sid_h) and the secret key sk_j which are sent thereto, the decryptor
- 10 3074 calculates tag ID information id_h which is decrypted from the privileged ID information sid_h' ($id_h = (id_h \cdot pk_j^{r+r'}) / (g^{r+r'})^{sk_j}$) (step S569). It is to be noted that the index “ sk_j ” appearing in this calculation formula refers to “ sk_j ”. The calculated tag ID information id_h is sent to the communication section 3072, which then transmits it to the client apparatus 3020 through the network 3080
- 15 (step S570). The client apparatus 3020 receives the transmitted tag ID information id_h at its communication section 3021 (step S571), and utilizes this tag ID information id_h in its subsequent inquiry to the backend apparatus 3050.

[0240] <features of embodiment 25>

- 20 In this embodiment, an arrangement is made that during re-privileging processing, a digital signature $\sigma' = E_{sk}(g^{r+r'} \mid id_h \cdot pk_j^{r+r'} \mid kid_j)$ is generated in the security server 3360, and this digital signature σ' is verified by the security server 3370 during the decrypting processing. Accordingly, during the decrypting processing, the authenticity of privileged ID
- 25 information which is re-privileged can be verified by the digital signature, allowing a storage of a wrong privileged ID information in the tag device 3310 to be detected in a more reliable manner.

[0241] While the digital signature σ' is generated in the security server 3360 in this mode, it is to be noted that the security server 3370 or a server of an authentication organization or the like may act for the generation of the digital signature σ' .

5 [Embodiment 26]

This embodiment is a modification of the embodiment 23, and differs from the embodiment 23 in that of a variety of information which constitute a tag ID information, only that information which is inherent to each tag device is privileged to provide privileged ID information. A distinction over the
10 embodiment 23 will be principally described below while omitting a description for matters which are common with the embodiment 23.

[0242] Fig. 58 is an illustration of a functional arrangement of a tag device 3410 of the present embodiment. It is to be noted that in Fig. 58, functional arrangements which are common with the embodiment 23 are designated by
15 like characters as used in the embodiment 23.

<an overall arrangement/hardware arrangement>

This remains to be similar to the embodiment 23 except that the tag device 3010 is replaced by a tag device 3410.

<pre-processing>

20 A distinction over the embodiment 23 resides in that of a variety of information which constitute tag ID information, only that information which is inherent to each tag device is privileged to provide privileged ID information sid_h . When using tag ID information having a data construction as illustrated in Fig. 52, the serial code (sc) 3205 represents information which is
25 inherent to each tag device, and privileged ID information will be $sid_h=(g^r, sc_h \cdot pk_j^r)$. This privileged ID information ($sid_h=(g^r, sc_h \cdot pk_j^r)$) is stored in a rewritable region 3411b of a confidential value memory 3411 of the tag

device 3410. It is also a distinction over the embodiment 23 that information which is common to products such as the version code (vc) 3202, manufacturer code (mc) 3203 and products code (pc) 3204 are encrypted ($E(vc)$, $E(mc)$, $E(pc)$) and stored in a read-only region 3411a of the

5 confidential value memory 3411. It is to be noted that a probability encryption or the like is used for encrypting information such as version code (vc) 3206 or the like which is common to every kind of products so that a different encrypted text may be obtained for the same products.

[0243] <processing>

10 Privileged ID decrypting processing and privileged ID updating processing of this example remain similar to the embodiment 23 except that the privileged ID information is represented by $sid_h = (g^r, sc_h \cdot pk_j^r)$. It is also another difference from the first mode that a read/write section 3012 extracts $E(vc)$, $E(mc)$, $E(pc)$ or the like from the read-only region 3411a of the
15 confidential value memory 3411 to be transmitted to a backend apparatus 3050 through its interface 3013 and client apparatus 3020 or the like as required for an inquiry to the backend apparatus 3050.

[0244] <features of embodiment 26>

20 In the present embodiment, since only information which is inherent to each tag device is privileged to provide privileged ID information, an amount of data which is subject to the privileging processing can be reduced to reduce the amount of calculation and amount of communication in comparison to an arrangement in which information which is common to each kind of products is also privileged to provide privileged ID information.

25 It should be understood that the present embodiment is not limited to the described modes or embodiments mentioned above. By way of example, the present invention can be carried out in a mode which comprises a

combination of embodiments, and in addition, a variety of processings mentioned above are not limited to those which are executed in a time sequence as described, but may be executed concurrently or individually depending on the processing capacity of the apparatus which executes a
5 processing or as required. In addition, it should be understood that modifications can be suitably made without departing from the spirit of the present invention.

[0245] When the described arrangements are implemented using a computer, a processing content of a function which is to be performed by each apparatus
10 is described in terms of a program. By executing the program on a computer, the processing function is implemented by the computer.

The program which describes such processing content can be recorded in a record medium which is readable by the computer. A record medium which is readable by the computer may comprise a magnetic recorder,
15 an optical disc, a magneto-optical record medium, a semiconductor memory or the like, for example. Specifically, by way of example, the magnetic recorder may comprise a hard disc unit, a flexible disc, a magnetic tape or the like; an optical disc may comprise DVD (Digital Versatile Disc), DVD-RAM (Random Access Memory), CD-ROM (Compact Disc Read Only Memory),
20 CD-R (Recordable) / RW (ReWritable); the magneto-optical record medium may comprise MO (Magneto-Optical disc) or the like; and the semiconductor memory may comprise EEP-ROM (Electrically Erasable and Programmable-Read Only Memory) or the like.

[0246] The distribution of the program may take place through a sale, a
25 transfer or a lease or the like of a portable record medium such as DVD, CD-ROM or the like in which the program is recorded. Alternatively, the program may be stored in a memory of a server computer, and may be

transferred from the server computer to another computer for purpose of distribution.

A computer which is used to execute the program may have the program which is recorded in the portable record medium or the program which is transferred from the server computer once stored in its own memory. When the processings are to be executed, this computer can read the program stored in its own record medium and executes processings in accordance with the program which is read out. In another mode of executing the program, the computer may directly read the program from the portable record medium, and then execute processings in accordance with the program. In addition, each time a program is transferred from the server computer to this computer, the computer may execute processings in accordance with the program received in a sequential manner. As a further alternative, rather than transferring the program from the server computer to this computer, the described processings can be executed by a so-called ASP (Application Service Provider) service in which the processing functions are realized by merely commanding execution and acquiring results. It is to be understood that a program in this mode is intended to include information which is used in a processing by an electronic computer and which conforms to a program (such as data which does not directly command a computer, but which has the property to rule the processing by the computer).

AVAILABILITY OF USE IN INDUSTRY

[0247] In accordance with the present invention, a likelihood that a distribution process of tag devices may be traced from information delivered from tag devices in RFID, for example, can be suppressed.